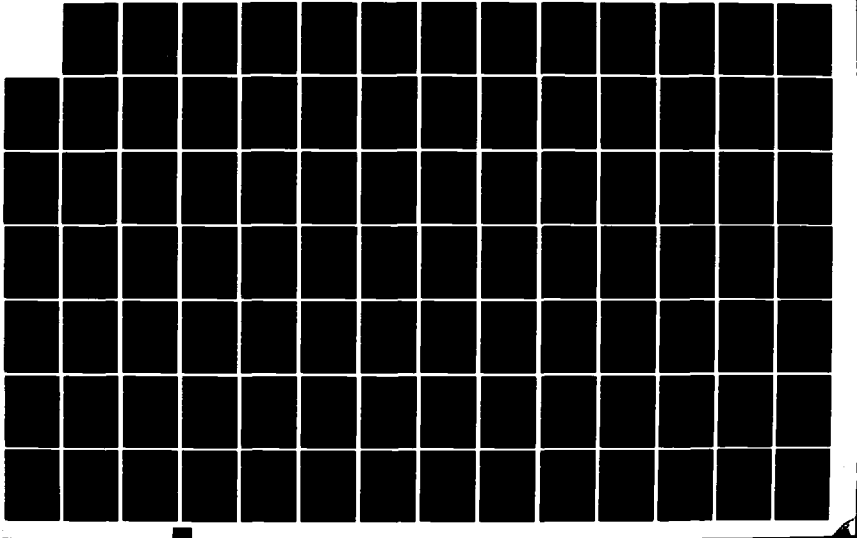
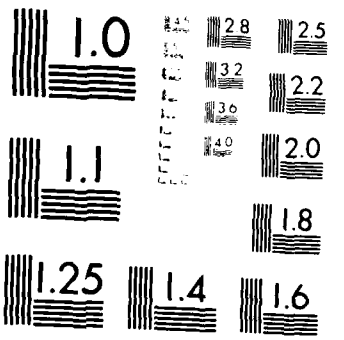
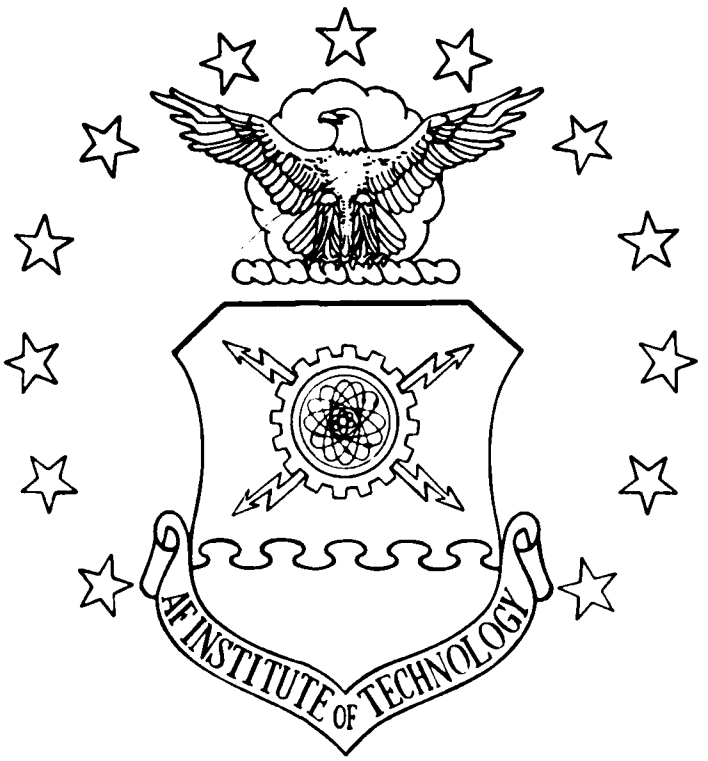
AD-A151 844 DEVELOPING AN AUTOMATED TECHNIQUE FOR TRANSLATING A RELATIONAL DATABASE I..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. E Y ALY 1/2

UNCLASSIFIED DEC 84 AFIT/GCS/MATH/84D-4 F/G 9/2 NL







DEVELOPING AN AUTOMATED TECHNIQUE FOR

TRANSLATING A RELATIONAL DATABASE INTO

AN EQUIVALENT NETWORK ONE

THESIS

Elsayed Yaser Aly
Maj., Egyptian Army

AFIT/GCS/MATH/84D-4

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DEVELOPING AN AUTOMATED TECHNIQUE FOR TRANSLATING A RELATIONAL DATABASE INTO AN EQUIVALENT NETWORK ONE

THESIS

Elsayed Yaser Aly
Maj., Egyptian Army

AFIT/GCS/MATH/84D-4

AFIT/GCS/MATH/84D-4

DEVELOPING AN AUTOMATED TECHNIQUE FOR TRANSLATING A RELATIONAL DATABASE INTO AN EQUIVALENT NETWORK ONE

THESIS

Presented to the Faculty of the School of Engineering

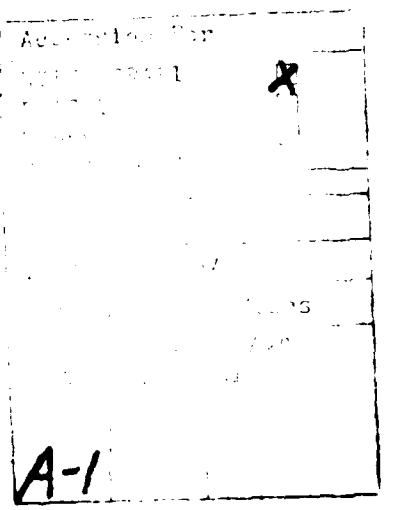
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science



by

Elsayed Yaser Aly

Maj., Egyptian Army

December 1984

Preface

The purpose of this study was to develop an automated technique for translating a relational database into an equivalent network one. This technique is the major part in creating a multi-model database management system from a network one and that will enable the users to implement both relational and network databases via one database management system.

The developed technique was used to translate a relational schema into a network one. The generated network schema had the optimum number of record types and sets, had no redundancy, and preserved the functional dependencies of the relational schema.

In developing the automated technique and writing this thesis, I have had a great deal of help from others. I am deeply indebted to my thesis advisor, Dr. Henry Potoczny, for his continuing patience and assistance in times of need. I also wish to thank Dr. Gary Lamont and Dr. Thomas C. Hartrum for reviewing my thesis and for their assistance and suggestions.

Finally, I wish to express my sincere gratitude to my father who taught me how to do everything as perfect as I can and encouraged me to continue learning all my life. Also, I wish to thank my wife Samya for her understanding and concern on these many nights when I was tied to my desk or my computer with work.

Table of Contents

	Page
Preface	ii
Abstract	iv
I. Introduction	1
Background	1
Problem Statement	2
Scope	3
Objectives	4
Assumptions	5
Order of Presentation	6
II. Relational and Network Data Models	7
The Relational Model	7
The Network Model	13
Comparison Between Relational and Network Models	14
Multi-Model DBMS	18
III. Translation Algorithms	21
Algorithm I	21
Algorithm II	26
Solutions Comparison	36
IV. The Automated Technique	37
The Modified Algorithm	37
Software Development	39
Software Testing	43
Implementation	44
V. Conclusions and Recommendations	45
Conclusions	45
Recommendations	47
Bibliography	48
Appendix A: Structure Charts	50
Appendix B: Developed Automated Technique	55
Appendix C: Resulted Network Schema	95
Appendix D SADT Diagrams	101
Vita	113

Abstract

This thesis developed an automated technique for translating a relational schema into an equivalent network one. In addition to the usual translation applications, this technique is the major step towards developing the multi-model data base management system which enables implementing both relational and network databases via one system, allows the use of either relational or network commands, solves the problem of converting applications, and eases the communication between data base management systems.

Reviewing the previous efforts in that field, there are two algorithms that were developed for the same objective. Each one was discussed, explained, and used to solve an illustrative example. Both algorithms were compared and one of them was chosen as a basis for developing the automated technique.

The chosen algorithm was modified to suit the requirements, and analyzed and decomposed using the SADT. The automated technique was developed and coded in PASCAL. In its development, it was considered to be user friendly, and to produce the network schema in a suitable format. The developed technique was tested and implemented using a relational schema in the third normal form. The resulting network schema was efficient, nonredundant, and had the minimum and necessary number of record types and sets as it was supposed to be.

Several conclusions were reached during the course of study, and some recommendations were proposed for further study according to the assumptions listed initially.

I. Introduction

1.1 Background

In recent years there has been a rapid growth in the use of large database systems. The network model is now in wide-spread use. The relational model was introduced and rapidly developed in the past decade. Both have attracted the most attention among the various models of database systems. The relational model appears to be easier, simpler, to provide more data independence, and to be superior in terms of the logical description of databases, whereas the network model is more efficient of space and time, and provides a more natural structure for the modelling of the world (11). Both data models are useful and important but neither is best for all applications.

A DBMS is designed to handle the applications of only one database model. It presents the following difficulties (11:82):

- The suitable data model for an application may be different from the existing DBMS.
- It is difficult for a DBMS to access the data maintained by a different DBMS.
- Conversion of all applications may be necessary if a different DBMS is installed.

The term "multi-model DBMS" is used to identify that the DBMS can handle more than one database model. This type of DBMS eases many problems such as (22:82):

- The database administrator can choose the data model that suits his application and allows him to be more productive.
- It solves the problem of converting applications from one data model to another.
- It eases the communications between the different DBMSs.
- The database administrator can incorporate the advantages of both relational and network models by using the first one for describing the database and the second one for implementing it.
- The user can access the database using either relational or network commands.

To achieve a multi-model DBMS, it is required to provide the uni-model DBMS with an automated technique to translate both the data model schema and the data manipulation commands of one database model to another that suits the uni-model DBMS. For example, if the DBMS is a network one, the automated technique has to support the translation of a relational schema to an equivalent network one and the translation of the relational commands to the network commands.

1.2 Problem Statement

It seems reasonable to consider the problem of translating a relational model into a network one for the following reasons:

- The relational database enables us to incorporate the recent results in design theory such as the normal forms and the lossless join property. For this reason the relational model is widely used now more than the other data models.
- Large percentage of the existing DBMSs are network systems since the network model was developed earlier than the relational one.

- Other researchers (13;19) have considered the problem of translating a network model into a relational one.
- Another researcher (9) has considered the mapping between relational and network operators.

Translating a relational schema into an equivalent network one enables the database administrator to implement a relational database via a network DBMS. On the other hand, translating the relational data manipulation commands into network commands will provide the ability to handle the database using either the relational commands or the network commands. Both translations are necessary to achieve the multi-model DBMS.

The problem that will be discussed in this thesis is to develop an automated technique for translating a relational database into an equivalent network one.

1.3 Scope

Translating a relational database into an equivalent network one has to be done through the following steps:

1. If the relational database is implemented via a relational DBMS, it needs a program to read its structure.
2. If it is not in the third normal form, it needs to be converted to the third normal form using the algorithms in (16;20;21).
3. Translating the relational schema into an equivalent network one.
4. Implementing the network schema via a network DBMS.
5. Reading the data from the relational database and loading it in the appropriate record types in the network database.

Besides that, translating the relational commands to network commands is necessary to completely achieve the multi-model DBMS.

If the problem is that there is a designed relational schema in an organization which has only a network DBMS, the solution is to develop an automated technique to translate it to an equivalent network one in order to implement it via the existing DBMS. This is simpler and easier than to repeat the design process completely which is more expensive and needs a long time.

The scope of this thesis investigation is to develop an automated technique for translating a relational schema into an equivalent network one. This solves the above problem and at the same time it is the major step in translating a relational database to a network one.

1.4 Objectives

The objective of this thesis effort is to develop an automated technique for translating a relational schema into an equivalent network one. The following points will be considered during developing this automated technique:

1. It should be able to deal with a relational schema of a finite number of attributes and relations.
2. It should be user friendly. This means that the user will input the required data of the relational schema to the automated technique.
3. The output, which is the network schema, should be in an understandable form. It is difficult to give the output in a graphical form because the number of record types and the owner-member relationships between them vary from one schema to another. The automated technique should overcome this difficulty and output the

network schema in an easy to understand form.

4. The automated technique should provide an efficient network schema with no redundancy, with the minimum number of record types, and with the minimum number of owner-member sets that achieve the equivalence to the input relational schema.

This objective will be achieved through the following steps:

1. Studying the previous efforts in this area.
2. Choosing a suitable algorithm.
3. Developing the automated technique for translating a relational schema into a network one using the modified algorithm.
4. Testing the automated technique using a relational schema to be translated into a network one.

1.3 Assumptions

The first assumption is that the relational schema, that will be translated into an equivalent network one, is in the third normal form, have the lossless join property, and embody the functional dependencies as keys (10). There are many algorithms to generate such a relational schema (16;20;21).

The second assumption is that the automated technique will consider the translation process only. If the relational database is already implemented via a relational DBMS, then another software is responsible to read its structure and pass it to the translation software. Also, it is assumed that another software will take the network schema as its input and implement it via a network DBMS.

1.6 Order of Presentation

The necessary concepts of the relational and network data models will be discussed in the following chapter, that is Chapter II. Also, it includes a comparison between the two models and how to achieve the multi-model DBMS from either the relational or the network model.

Chapter III includes a summary of the previous work done in that field. There are two algorithms that were already developed for this purpose. Each one is discussed in detail and used to solve an example. Both solutions are compared and one algorithm is chosen for developing the automated technique.

In Chapter IV, the chosen algorithm is modified to be suitable for programming. Then, the modified algorithm is analysed and decomposed using the SADT (Structured Analysis and Design Technique), and the structure charts are drawn. The algorithm for each module is designed and the complete automated technique is developed. Also, the chapter includes the results of testing and implementing the automated technique.

The conclusions and recommendations are presented in Chapter V.

II. Relational and Network Data Models (4;7;10)

2.1 The Relational Model

2.1.1. Basic definitions. The basic component of a relational database is the relation which is in the form of a table. Each column in this table has a unique name which is called the attribute name. Each attribute has its domain. Any value for that attribute in the table is drawn from its domain. Each row of the table is called a tuple. It is an ordered set of attribute values and this set is unique. The relational scheme for a relation is the relation name and the attribute names. A relational database is a set of relations and a relational schema is the set of relational schemes corresponding to the given set of relations (10).

2.1.2 The Relational Operators. The relational operators are the project, select, and join. They are denoted by π, σ, and * respectively. The operands of these operators are the relation schemes. A relational expression is evaluated by substituting the relations for the relation schemes and applying the operators according to their definitions.

The select operator constructs a new table by taking a horizontal subset of the existing table, that is, all the rows that satisfy some condition. The project operator forms a vertical subset of an existing table by extracting specified columns and removing any redundant rows in the extracted table. If two tables each have a column defined over some common domain, they may be joined over those two columns. The result of the join is a new table in which each row is formed by concatenating two rows, one from each table, such that the two rows have the same value in those two columns (4).

2.1.3 Types of Dependency.

(a) Functional Dependency

In a relation R, attribute Y is functionally dependent on attribute X if and only if, whenever two tuples of R agree on their X-value, they also agree on their Y-value. This is denoted by X----->Y. Attribute Y is fully functionally dependent on attribute X if it is functionally dependent on X and not functionally dependent on any proper subset of X.

(b) Multivalued Dependency

Given a relation R with attributes A, B, and C, the multivalued dependency A --> -->B holds if and only if the set of B-values matching a given (A-value, C-value) pair in R depends only on the A-value and is independent of the C-value. A, B, and C may be composite. Multivalued dependency is denoted by MVD.

(c) Join Dependency

Relation R satisfies the join dependency (X, Y, ..., Z) if and only if it is the join of its projections on X, Y, ..., Z where X, Y, ... Z are subsets of the set of attributes of R. It is denoted by JD and written ⋈ (X, Y, ...Z) (7).

2.1.4 Normal Forms (4;7;10)

A relation R is in the first normal form (1NF) if and only if all underlying domains contain atomic values only. It is in the second normal form (2NF) if and only if it is in 1NF and every nonkey attribute is fully dependent on the primary key. If it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key, then the relation R is in the third normal form (3NF).

A relation R is in the Boyce/Codd Normal Form (BCNF) if and only if every determinent is a candidate key. Determinent is an attribute on which some other attribute is fully functionally dependent.

A relation R is in the fourth normal form (4NF) if and only if, whenever there exists MVD in R, say A --> --> B, then all attributes of R are also functionally dependent on A. It is in the fifth normal form (5NF) if and only if every join dependency in R is implied by the candidate key of R.

The different types of normal forms can be obtained for a given relation using the nonloss decomposition technique. Starting with an original relation and a set of constraints, the relation can be reduced to a collection of relations that are equivalent to the original one and in some way preferable to it. The constraints are used as a guide during the decomposition process. This process can be done as follows:

- The projections of the 1NF relation to eliminate any non full FDs will produce 2NF relations.
- The projections of 2NF relations to eliminate any transitive dependency will produce 3NF relations.
- The projections of 3NF relations to eliminate any remaining FDs in which the determinent is not a candidate key will produce a collection of BCNF relations.
- The projections of these BCNF relations to eliminate the MVDs that are not also FDs will produce a collection of 4NF relations.
- The projections of these 4NF relations to eliminate the JDs that are not implied by candidate keys will produce a collection of 5NF relations.

The objective of this reduction process is to reduce the redundancy and to avoid problems in the updating operations.

2.1.5 Keys

The primary key of a relation R is an attribute or a set of attributes with values that are unique within the relation and thus can be used to identify the tuples of the relation. If the relation has more than one attribute combination possessing the unique identification property, they are called candidate keys. A candidate key that is not the primary key is called an alternate key. An attribute is sometimes called a foreign key in a relation if its values are values of the primary key in another relation.

There is another definition for the key of a relation with functional dependencies (7). If R is a relation scheme with attributes $A_1A_2...A_n$ and functional dependencies F, and if X is a subset of the attributes, then X is a key of R if:

1. X --> $A_1A_2...A_n$ is in the closure of F(F+).
2. For no proper subset Y $\subset$ X is Y --> $A_1A_2...A_n$ is in F+.

The first condition implies that the key is unique within the relation. The second one implies the minimality of the key, i.e., the key is the minimal set of attributes that functionally determines all attributes.

2.1.6 Armstrong's Axioms (7)

Suppose R is a relation schema and A, B, and C are some of its attributes. Suppose that the functional dependencies A --> B and B --> C are hold in R. Then A --> C must hold in R. So A --> C is logically implied by the set of functional dependencies F for a relation R. Let F+, the closure of F, be the set of functional dependencies implied by F, i.e.,

$$F+ = \{X \dashrightarrow Y \mid F \models x \dashrightarrow y\}$$

Computing F+, the closure of F, is essential and important to determine keys, to understand logical implications among F, and to know whether a particular functional dependency is logically implied by F.

Armstrong's axioms are the set of rules that allows us to deduce all the functional depencencies in F+. This set of rules is complete and sound. Complete means that it allows us to deduce all the FDs in F+ and sound means that if any FD is not in F+, then it is impossible to deduce it from F using this set of rules.

Assume that there is a relation scheme with set of attributes U, the universal set of attributes, and a set of FDs F involving only attributes in U. The set of rules are:

1. Reflexivity: If $Y \subseteq X \subseteq U$, then $X \dashrightarrow Y$ is logically implied by F.
2. Augmentation: If $X \dashrightarrow Y$ holds, and $Z \subseteq U$, then $XZ \dashrightarrow Y z$ holds.
3. Transitivity: If $X \dashrightarrow Y$ and $Y \dashrightarrow Z$ hold, then $X \dashrightarrow Z$ holds.

There are other rules that follow from Armstrong's axioms. They are:

1. Union: If $X \dashrightarrow Y$ and $X \dashrightarrow Z$ hold, then $X \dashrightarrow YZ$ holds.
2. Pseudotransitivity: If $X \dashrightarrow Y$ and $WY \dashrightarrow Z$ hold, then $XW \dashrightarrow Z$.
3. Decomposition: If $X \dashrightarrow Y$ holds and $Z \subseteq Y$, then $X \dashrightarrow Z$ holds.

2.1.7 Computing Closures (7;10)

In general, computing F+ for a set of functional dependencies F is a time consuming task because the set of dependencies in F+ can be large if F is small. Consider the set $F = \{A \dashrightarrow B_1, A \dashrightarrow B_2, \ldots, A \dashrightarrow B_n\}$, then F+ includes all the dependencies $A \dashrightarrow Y$ where Y is a subset of $\{B_1, B_2, \ldots, B_n\}$. As there are 2^n subsets, F+ is large even for reasonable F.

But computing X+ for a set of attributes X is not hard. It takes time proportional to the length of all the dependencies in F written out. Also, telling whether X --> Y is in F+, is not harder than computing X+.

Given a finite set of attributes U, a set of FDs F on U, and a set X $\subseteq$ U, the following algorithm computes X+.

Algorithm 2.1

1. X+ = X
2. While F includes a FD Y --> Z such that Y $\subseteq$ X+ and Z $\not\subseteq$ X+ do

   X+ = X+Z; (X+ concatenated with Z)

2.1.8 Covers of Sets of Dependencies

Let F and G be two sets of FDs. F and G are equivalent, written F $\equiv$ G, if F+ = G+. If F and G are equivalent, then F covers G and G covers F. The following algorithm determines if F and G are equivalent.

Algorithm 2.2

1. For each dependency Y --> Z in F do

   Compute Y+ using G

   If Z $\subseteq$ Y+ then Y --> Z is in G+

   else F $\not\equiv$ G, stop
2. For each dependency Y --> Z in G do

   Compute Y+ using F

   If Z $\subseteq$ Y+ then Y --> Z is in G+

   else F $\not\equiv$ G, stop

A set of functional dependencies F is minimal if:

1. Every right side of a FD in F is a single attribute.
2. For no X --> A in F is the set F- {X --> A} $\equiv$ F.
3. For no X --> A and proper subset Z of X is

   F - {X --> A} $\cup$ {Z --> A} $\equiv$ F

The second condition guarantees that no dependency in F is redundant. The third one guarantees that no attribute on any left side of a dependency is redundant. The following algorithm can be used to minimize the number of dependencies in F, that is, to produce the minimal F. Each part of this algorithm corresponds to one of the previous three conditions.

Algorithm 2.3

1. For each dependency $Y \rightarrow Z_1Z_2...Z_n$ do

   Add $Y \rightarrow Z_1$, $Y \rightarrow Z_2$, ..., $Y \rightarrow Z_n$ to F

   delete $Y \rightarrow Z_1Z_2,...Z_n$ from F.

2. For each dependency $Y \rightarrow Z \subseteq F$ do

   Compute $X = (F - \{Y \rightarrow Z\})+$

   If $Y \rightarrow Z \subseteq X$

   Then delete $Y \rightarrow Z$ from F.

3. For each dependency $Y \rightarrow Z \subseteq F$ do

   For each attribute A in Y do

   If $\{Y-A \rightarrow Z\} \subseteq F+$

   Then Add $Y-A \rightarrow Z$ to F

   delete $Y \rightarrow Z$ from F.

2.2 The Network Model

The network model consists of two elements only. One is the record type and the other is the set or link between record types. A network schema is represented as a directed acyclic graph with the nodes corresponding to record types and edges corresponding to sets or links. An edge is directed from a record type, called the owner, to another record type, called the member, and represents a many-to-one relationship. An owner

record may have many member records in its set, but a member record has only one owner record of a particular set. Record types which have no owner must be owned by the system.

The record type is a collection of a number of data items. The values that a data item assumes are called its occurences. The occurence of a record is any combination of the occurences of its constituent data items.

There are two kinds of operations for the network model. First, there are selections on logical record types which are similar to the selections on relational model. The other type of operations is the following of links in one direction or the other.

2.3 Comparison Between Relational and Network Models (4;6)

In this section, the relative merits of the relational and network approaches will be discussed as a basis for the conceptual view of the system. The comparison will be concentrated on these approaches only because the hierarchies are considered as a restricted form of the networks and because the relations and networks are in direct competition in the database field. Although the discussion is concentrating mainly on the conceptual level, several of the points are also related to the external level.

2.3.1. The Conceptual Model

The conceptual schema should serve as a stable base for the overall system. It should not be dependent on the characteristics of any DBMS so that it can handle the replacement of one DBMS by another. It should not have to be changed unless some adjustment in the real world requires some definition to be adjusted too, so that it may continue to

reflect reality. One example of adjustment is the expansion of the conceptual schema to reflect a larger portion of reality.

The most important properties of the conceptual view are the following:

1. It should be easy to understand and to manipulate.

There are many aspects that help in achieving an easy to understand conceptual view. Some of these aspects are:

- The number of basic constructs should be small.
- Distinct concepts should be clearly separated.
- Symmetry should be preserved.
- Redundancy should be carefully controlled.

The aspects that help the conceptual view to be easy to manipulate are:

- The number of operator types should be small.
- Very high-level operators should be available.
- Symmetry should be preserved.

2. It should have a good theoritical base.

It is essential that the conceptual level be founded on a solid base of theory. Its behavior must be known clearly and must be consistent with the user expectations. The permitted and the prohibited must be fully determined. Also, it is important to know all the problems that are expected to occur.

2.3.2. The Relational Model

There is no doubt that relations are easy to understand. It uses one basic data construct, that is the relation. Most of the researches in security and integrity have taken the relational model as a starting

point because it provides a clean conceptual view. Also, the relational model seems to meet the symmetry and non redundancy aspects. The normalization guarantees the nonredundancy in the conceptual schema.

Relations are also easy to manipulate; very high-level operators as well as more familiar low-level operators are available. The high-level operators are those of the relational algebra and equivalent languages. The number of distinct operators in any language is very small because there is only one data construct to deal with. Also, the relational languages provide the symmetric exploitation which is the ability to access a relation by specifying known values for any combination of its attributes, seeking the values for its other attributes.

The relational approach is based on mathematical set theory. Also, normalization theory provides a set of guidelines for the design of a relational schema. The theory of relational completeness provides a valuable tool for measuring the power of a language and for comparing different languages.

Most relational data manipulation languages are nonprocedural, that is, the programmer specifies what results are required without specifying how the system is to access the database. The DBMS is responsible for choosing the appropriate access paths. Also, nonprocedural languages permit accessing many records for each command. In addition to that, one relational data manipulation command can be constructed by combining other commands. Relational commands can be used to define the database views.

2.3.3. The Network Model

Networks are somewhat less easy to understand than relations. In DBTG, there are five data constructs: record type, DBTG set, singular set, ordering, and repeating group. A more complex problem in the networks is that the DBTG set construct bundles together at least three distinct concepts. These concepts are:

- It carries information, namely, the association between the two record types involved.
- It provides an access path.
- It represents certain integrity constraints.

A network schema involving sets has less symmetry of representation than an equivalent relational schema, since some information is represented as records and some as sets. A network schema can be just as redundant as an equivalent relational schema if it does not involve any inessential sets.

Networks are less easy to manipulate than relations. Each data construct needs its own set of operators. So the networks require more operators than relations. Also it needs more integrity constraints.

There is no theory to assist with the design of a network schema that is as complete as the normalization theory is for relations. Normalization theory can be applied to the records of the network but only after a decision has been made as to which information is to be represented by records and which by other means.

The CODASYL network data manipulation language is a procedural one because the programmer has to specify how the DBMS is to access the database. This language permits the programmer to access a

single record for each command. Then, he may access either another record within the same record type or a record of a different type within the same set as the current record. Also, it permits the programmer to take advantage of the network structure by writing efficient application programs using a sequence of commands to specify how the DBMS is to access the data.

2.4 Multi-Model DBMS

A multi-model DBMS must meet two main requirements: it must support both relational and network schemas, and it must support both procedural and nonprocedural languages. There are two main approaches to achieve that: the mapping approach and the composite approach (11).

The mapping approach, illustrated in fig. 2.1., suggests that a schema for one data model can be generated from the schema of the other data model. Data manipulation commands against the generated schema are translated to equivalent commands which can be processed against the original schema. A DBMS in which a relational schema can be mapped to an underlying network schema is an example of the mapping approach.

The composite approach, illustrated in fig. 2.2, suggests that a schema for one data model be embedded into schema of the other model. The resulting common schema contains objects which can be viewed as both relational and network objects. Relational and network data manipulation commands use the common schema.

From the user's prospective, there is no main difference between the two approaches. But in the mapping approach, a program is restricted to one type of command while in the composite approach, a program can mix network and relational commands.

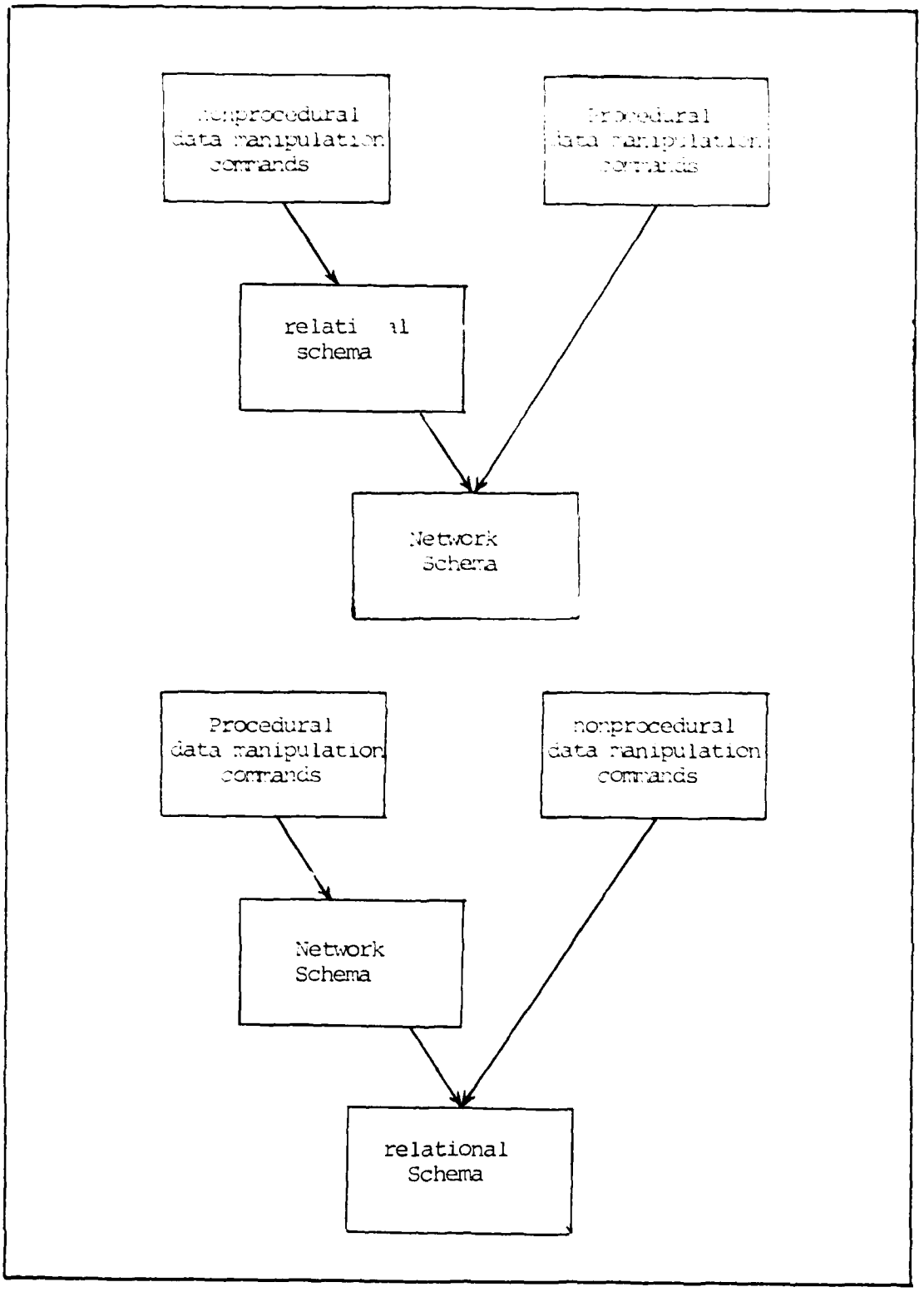


Fig. 2.1. The mapping approach

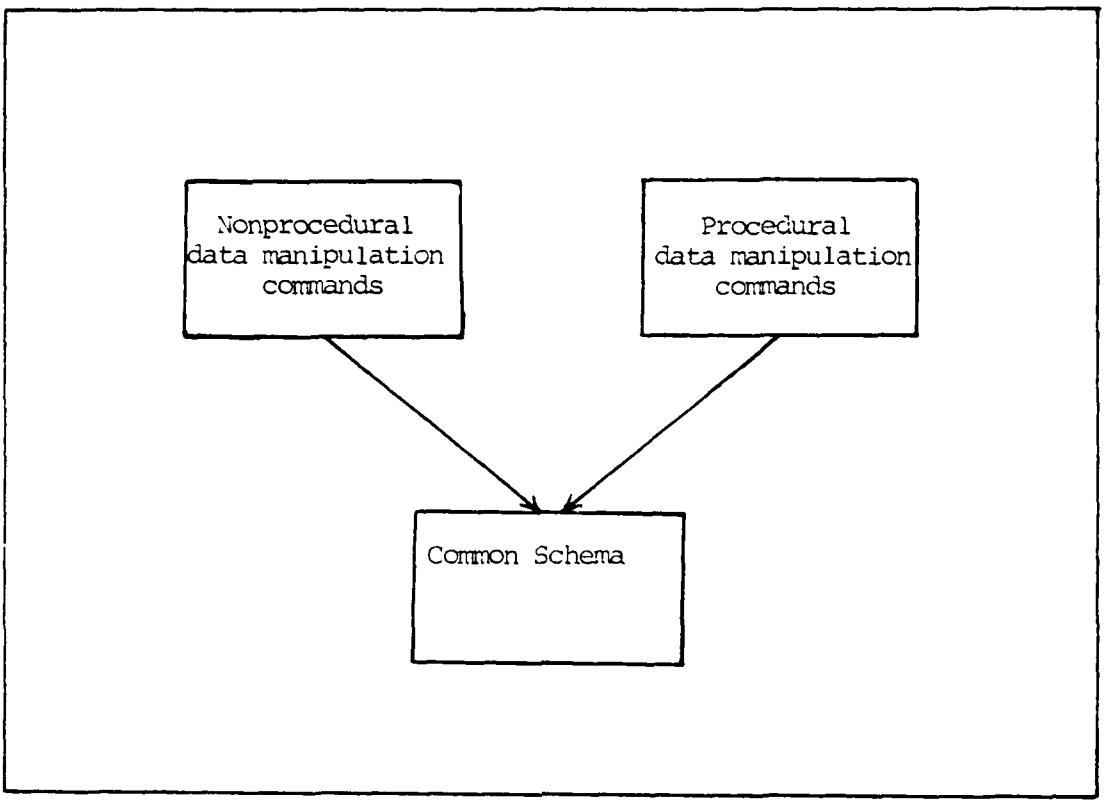


Fig 2.2. The Composite Approach

From this brief discussion of both the mapping and composite approaches, it is clear that the mapping approach is the suitable one for the objective of this thesis. A relational schema can be translated into an equivalent network one, and implemented via a network DBMS. Translating the relational data manipulation commands into network commands which is already discussed in some researches (9), will achieve the multi-model DBMS.

III. Translation Algorithms

There are two algorithms for translating a relational schema into an equivalent network one. These algorithms were developed by previous researchers. In this chapter, each algorithm is explained in detail, and used to solve an example. This example is chosen such that it illustrates every step in both algorithms. The two solutions are compared and one algorithm is selected as a basis for developing the required automated technique. It is important to mention that no previous researches attempted to develop such an automated technique.

Both the algorithms start with a relational schema that is synthesized from the attributes and functional dependencies using the algorithms in (16;19;20) which produce relational schemes in the third normal form that embody the FDs as keys and have the lossless join property.

The example is: Given the following relational schems, generate the equivalent network schema. Keys are underlined.

R1(ABC), R2(HDA), R3(DK), R4(KIA)

R5 (PHI), R6(KJB), and R7(PJ).

3.1 Algorithm I.

3.1.1. Algorithm (10;14)

The steps of this algorithm are:

1. Create a record type N_i for each relation schema R_i.
2. For each pair of record types $N_i, N_j (i \neq j)$

   create a set such that N_i owns N_j if R_j contains a key of R_i.

3. For each pair R_i, R_j such that

   (i) $R_i \subseteq R_j+$

   (ii) There is no directed path from N_i to N_j.

   (iii) There is no R_k such that $R_i \subseteq R_k+ \subseteq R_j+, K \neq j$

   create a set with N_i owns N_j.

4. For every N_i, remove all the attributes that appear in an ancestor of N_i.

5. While a new record type is created do

   If $N_{ki},\ldots,N_{km}$ $(m>1)$ are record types and X is the set of attributes that appear in every N_{kj} and x is in no other record types, then create a new record type N_k consisting of the attributes of x and make N_k the owner of every NK_j. Delete X from N_{kj}. Create $R_k(X)$ corresponding to N_k.

6. For every N_i, choose one key x of R_i such that all attributes contained in X are also in N_i (not every N_i has such an X). Make X a key of N_i with duplicates allowed.

7. For every N_i that is not a member in any set, create a system owned set with N_i as the member.

8. If $R_i \subseteq R_j+$ and there is no link from N_i to N_j, then create a set with N_i as owner and N_j as member. (optional)

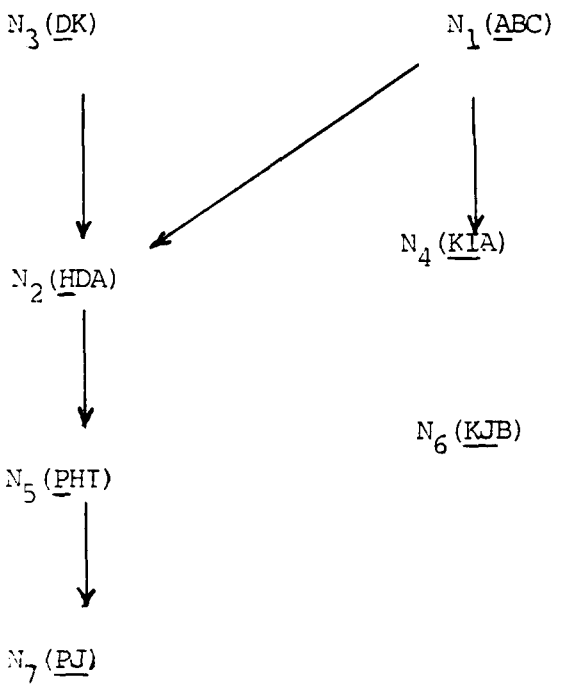
9. Optionally create additional system owned sets.

10. Search/sort keys with duplicates may be created freely for any set in the network.

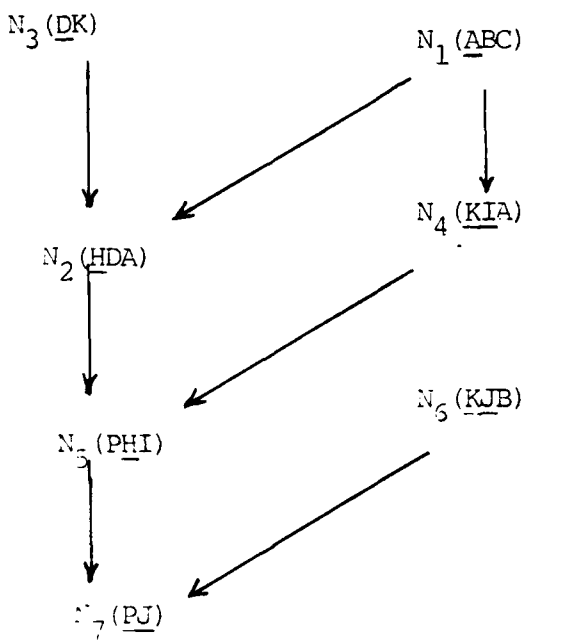
3.1.2 Solution of the Example:

Step 1: Create a record type N_i for each relation schema R_i i=1,2,...,7.

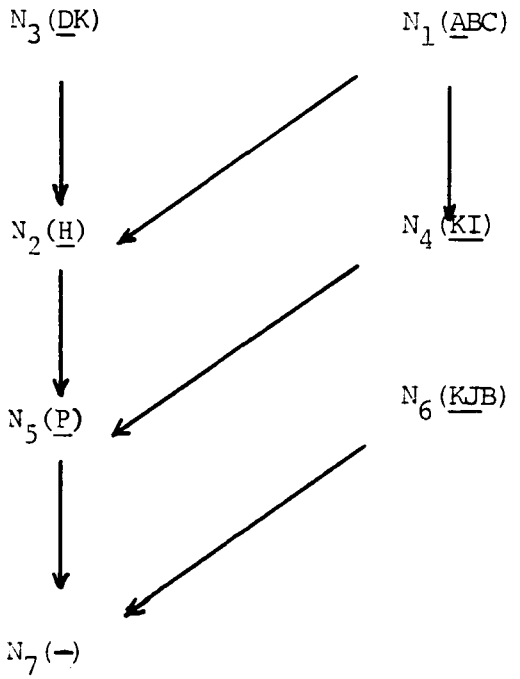
Step 2:



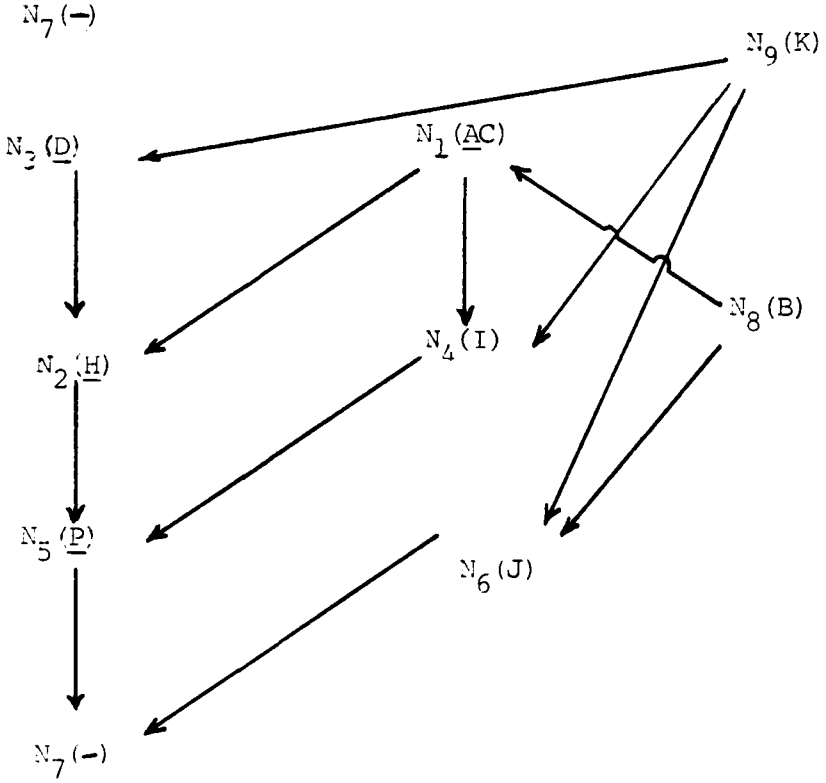
Step 3:



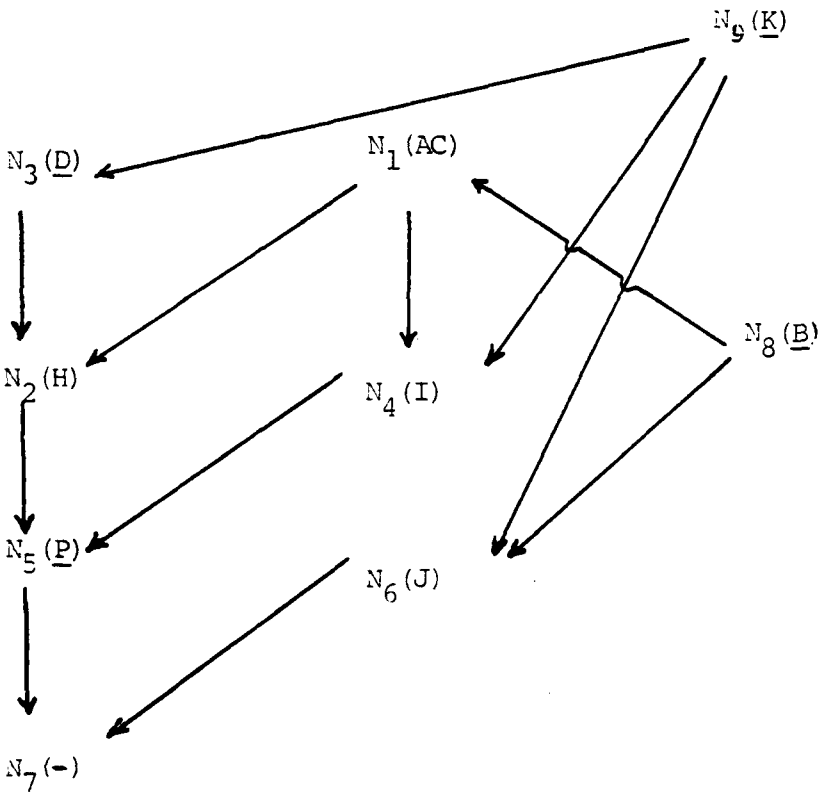
Step 4:


N_3 (DK)
N_1 (ABC)
N_2 (H)
N_4 (KI)
N_5 (P)
N_6 (KJB)
N_7 (–)

Step 5:


N_9 (K)
N_3 (D)
N_1 (AC)
N_8 (B)
N_4 (I)
N_2 (H)
N_5 (P)
N_6 (J)
N_7 (–)

Step 6:



Step 7:

Since N_8 and N_9 are not members in any set, create a system owned set for each.

3.2 Algorithm II

To understand this algorithm, it is important to explain two special functions that were defined by the author of the algorithm (12). These two functions are: MERGE and SPLIT. Each one will be defined and explained.

3.2.1 Special Functions

A. MERGE (S,N_1,N_2)

S, N_1, and N_2 are sets of attributes such that $S \subseteq N_1$ and $S \subseteq N_2$. N_1 and N_2 are in the form of networks, i.e., they each constitute the attributes of a network structure.

The operation MERGE puts together all occurences of S in N_1 with all occurences of S in N_2 without duplicating any value. The operation maintains the proper set relationships of the occurences being combined. In general, the S-owner sets and the S-member sets for S in N_1 may be different from those of S in N_2. Let O_i denote the collection of S-owner sets and M_i denotes the collection of S-member sets for S in N_i, i being either 1 or 2. MERGE operates in such a way that when it is done, the collections of S-owner and S-member sets become $O_1 \cup O_2$ and $M_1 \cup M_2$ respectively. It preserves all the relationships that were associated with S before the operation either in N_1 or in N_2.

B. SPLIT (S,N)

S and N are two sets of attributes of some relation R such that $S \subseteq N$. The operation separates S from s^1, where $s^1 = N-S$, and sets up the proper relationship between them. In effect, N is converted into a set with S and s^1 as the records of the set, except in some situations where two sets are generated and an additional dummy record is used. SPLIT (S,N) may result in the following types of sets:

(a) If S ---> s^1 and s^1 -/-> S,

the set is formed with owner s^1 and member S.

(b) If s^1 ---> S and S -/-> s^1,

the set is formed with owner S and member s^1.

(c) If S ---> s^1 and s^1 ---> S,

the set is formed with either S or s^1 as the owner and the other one as the member.

(d) If S -/-> s^1 and s^1 -/-> S, a dummy record is created, two sets are formed - one with owner S and the other with owner S^1, both having the dummy record as the member.

Actually, N might be included in one or more sets as a single record. SPLIT must maintain the proper set relationships for these sets.

3.2.2. Algorithm (12)

This algorithm has two parts: A and B. Each part will be discussed in detail.

(A) This part finds the common attributes from a given relation-attribute (RA) matrix and then finds the common attribute groups. The algorithm of this part is:

Begin

1. C <--- ϕ
2. For j=1 to n do

   Begin
3. find the no. of 1's in column j and call it COUNT
4. If COUNT > 1 then C = C U $\{A_j\}$

   end
5. K <--- 1
6. While C $\neq \phi$ do

   Begin
7. Select any attribute A_j from C
8. find the set of all attributes in C - $\{A_j\}$ where columns in the RA matrix are identical to column j
9. D_k <--- $\{A_j\}$ U S
10. D <--- C - C_k
11. K <--- K+1

   end;
12. G <--- $\{D_p \mid 1 \leq P \leq K-1\}$

   end;

Steps 2-4 pick up common attributes and steps 5-11 form the common attribute groups.

(B) This part forms the network schema from the given relational schema. The inputs to this part are the set of relations in 3NF, set of FDs, and the common attribute groups G generated from part A. A_{Ri} is the set of attributes for R_i i=1,2,...,m and at each stage N contains attribute groups processed so far. The algorithm of this part is

```
     Begin
1.    N <--- φ
2.    For i=1 to m do
        Begin
3.       Find all groups in G whose attributes are element of A_Ri and call
         the collection of these groups G_i
4.       Order G_i in such a way that the groups containing candidate keys,
         if any, are placed at the end.
5.       While G_i ≠ φ do
           Begin
6.         Select the group X that appears at the beginning of G_i
7.         G_i <--- G_i - X
8.         If X = A_Ri then
9.            if X∈N then MERGE (X,A_Ri,N)
10.                   else N <--- N ∪ X
11.           else Begin
12.                SPLIT (X,A_Ri)
13.                If X∈N then MERGE (X,A_Ri,N)
14.                       else N <---N ∪ X
                   end;
15.           A_Ri <--- A_Ri - X
           end (while)
        end (for)
     end;
```

3.2.3 Solution of the Example

(A) The relation-attribute (RA) matrix is

R \ A	A	B	C	D	H	I	J	K	P
R1	1	1	1						
R2	1			1	1				
R3				1				1	
R4	1					1		1	
R5					1	1			1
R6		1					1	1	
R7							1		1

C = {A, B, D, H, I, J, K, P}

G = {A, B, D, H, I, J, K, P}

(B)

G_1 = B,A

G_2 = D,A,H

G_3 = K,D

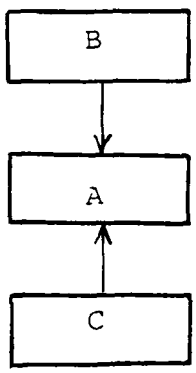
G_4 = A,K,I

G_5 = H,I,P

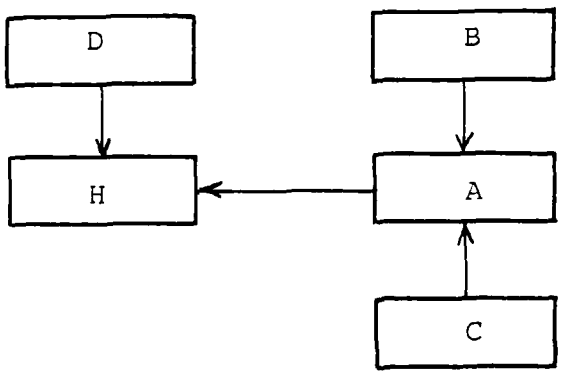
G_6 = B,K,J

G_7 = P,J

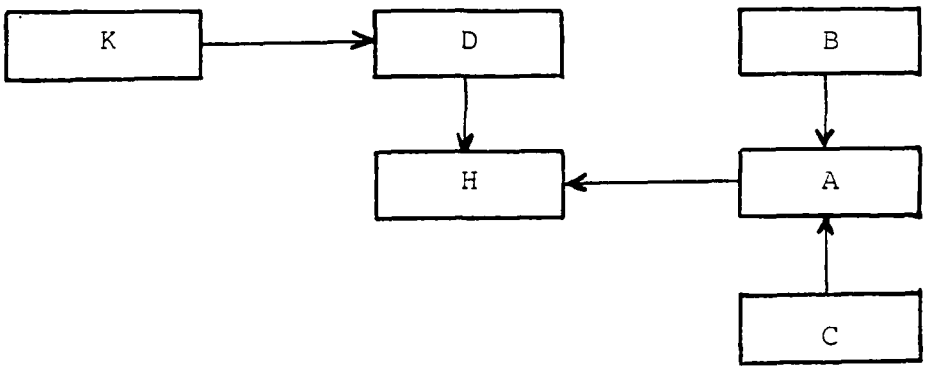
* N after processing G_1



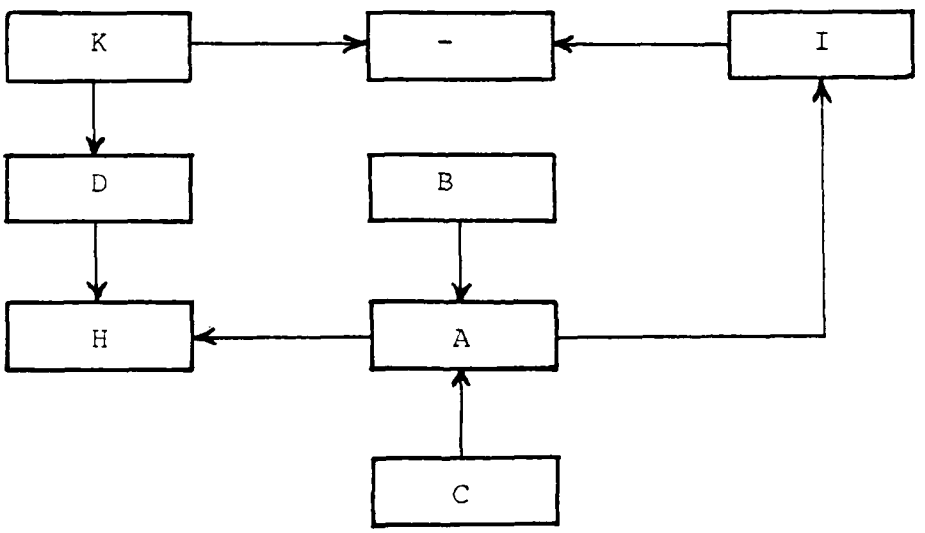
* N after processing G_2



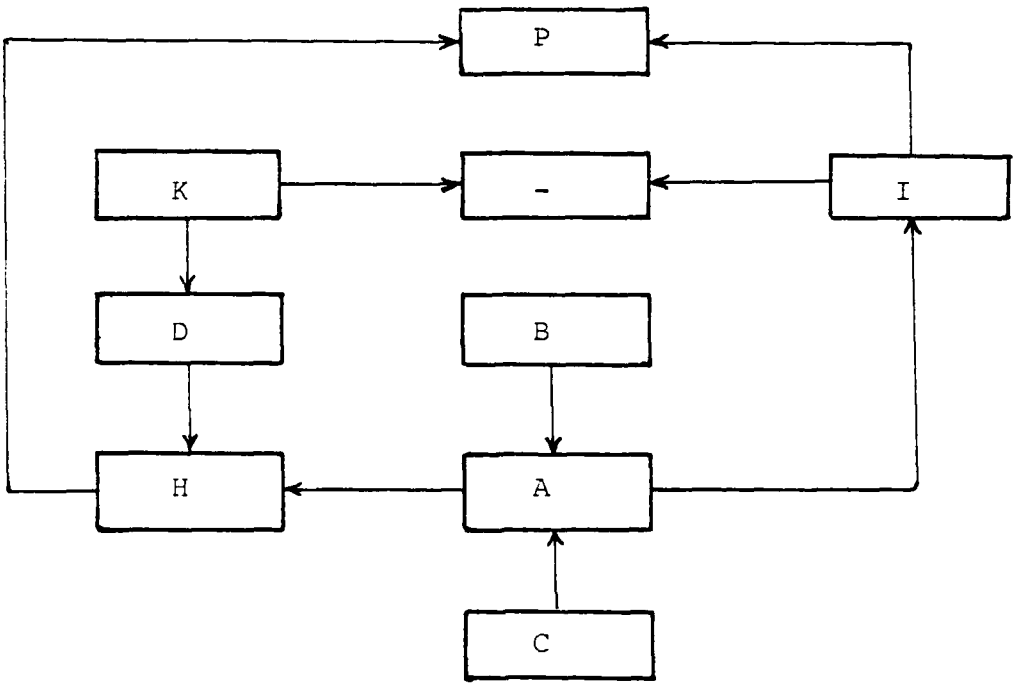
* N after processing G_3



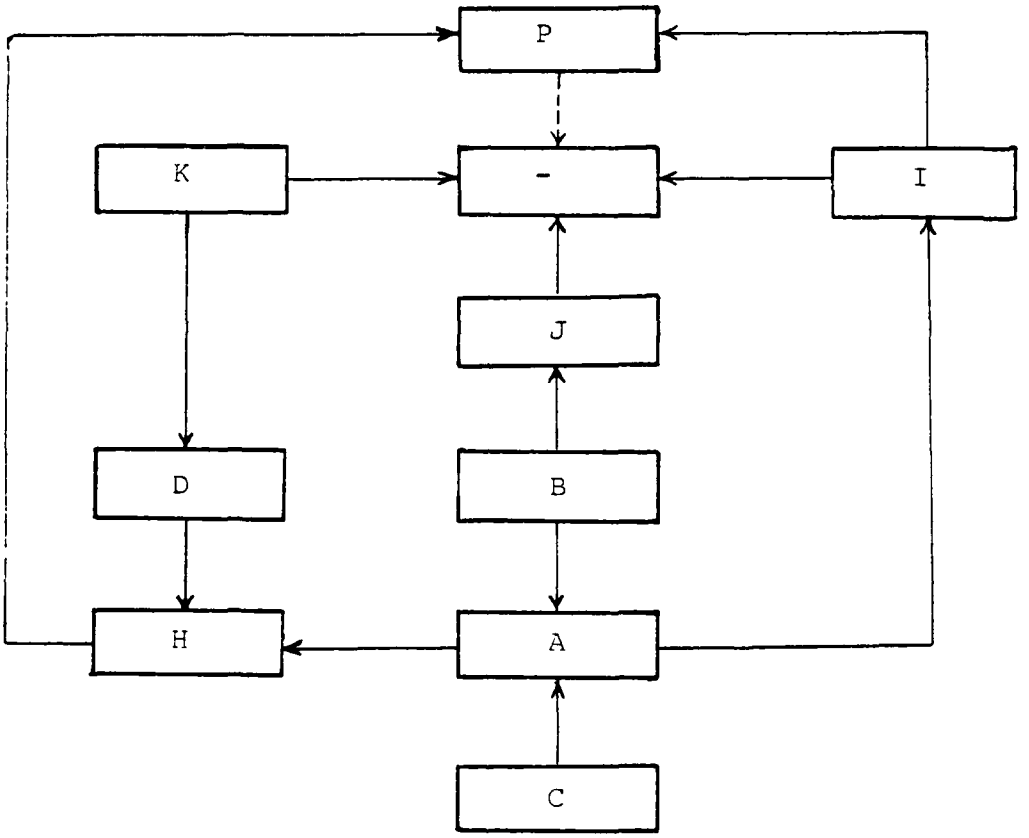
* N after processing G_4



* N after processing G_5



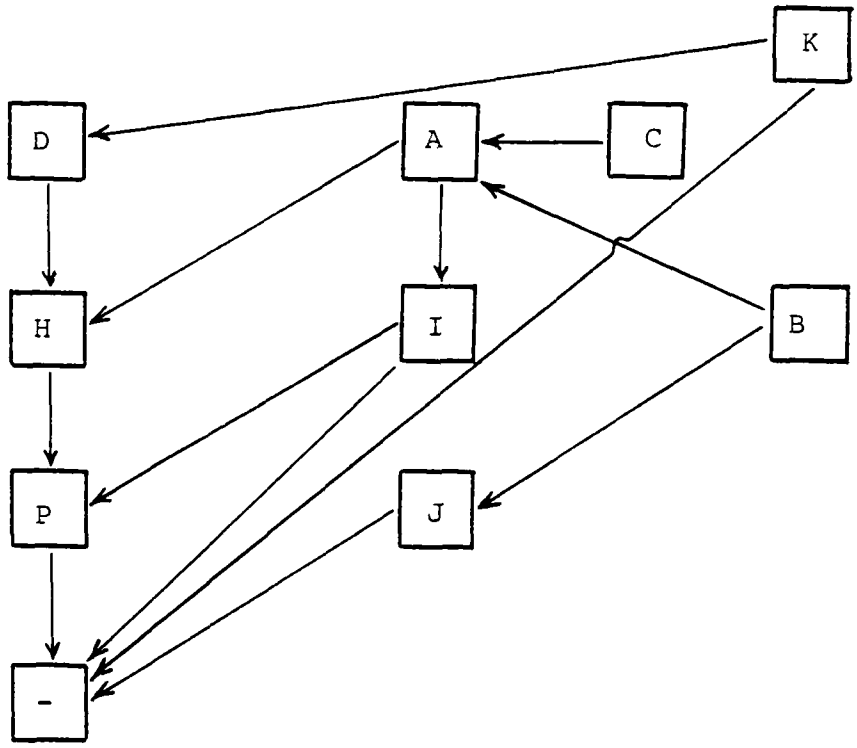
* N after processing G_6



* N after processing G_7

Only the set indicated by (--->) in the previous figure is added in this setp.

The final designed network is:



3.3 Solutions Comparison

Comparing the final solutions of algorithms I and II, it is clear that they are approximately similar. There are few minor differences between them but these differences do not affect any property of the designed network schema. These differences are illustrated in table 3.1.

No	Algorithm I	Algorithm II
1	A and C are in one record type	A and C each is in a separate record and A owns C in a set
2	K owns I and J, each in a set	Each of K, I, and J owns the dummy record in a set
3	The system owns K and B, each in a set	The system owns K, C, and B, each in a set
4	Number of record types is 9	Number of record types is 10
5	Number of sets is 12	Number of sets is 13
6	Time requirements is of $0(n^2)$	Time requirements is of $0(n^2)$
7	Space requirements is of $0(n)$	Space requirements is of $0(n)$
	n is the no. of relations	

Table 3.1. Comparison between the solutions

From the final solutions of both algorithms, and from table 3.1, the following points are noticed:

1. Both solutions have the same degree of nonredundancy and the same relationships between record types.
2. Both solutions have the same time and space requirements.
3. The network schema generated from algorithm I has a small number of record types and sets than that generated from algorithm II.
4. Steps of algorithm I are simpler and easier for coding than those of algorithm II.

For the above reasons, algorithm I is chosen for developing the required automated technique for translating a relational schema into an equivalent network one.

IV. The Automated Technique

4.1 The Modified Algorithm

After discussing algorithm I, which is chosen to develop the automated technique, and after using it to solve an example, the following remarks are appropriate:

1. Steps 1 to 7 are the essential steps in the algorithm and steps 8, 9, and 10 are optional steps to increase the efficiency of the generated network schema. These steps 8, 9, and 10, will not be considered in developing the automated technique only for the simplicity reasons.
2. Step 2 in the algorithm is redundant and is dominated by step 3 because if R_j contains a key of R_i then $R_i \subseteq R_j+$. So, step 2 can be removed from the algorithm.
3. Choosing the key, that is step 6, will be left for the database administrator because it is very difficult to deal with this problem in a computer program. The automated technique will try to keep the given keys of the relational schema as long as they are in the corresponding record types. But for the new record types, the database administrator will be responsible to choose their keys.

Applying the previous remarks to algorithm I, the following is the modified algorithm which will be used to develop the automated technique for translating a relational schema into an equivalent network one.

Algorithm

1. Create a record type N_i for each relation scheme R_i
2. For each pair R_i, R_j such that
   (i) $R_i \subseteq R_j^+$
   (ii) There is no directed path from N_i to N_j
   (iii) There is no R_k such that $R_i \subseteq R_k^+ \subseteq R_j^+, K \neq j$

   Create a set with N_i as owner and N_j as member.
3. For every N_i, remove all the attributes that appear in its ancestors.
4. For every attribute in the given set of attributes do

   If the attribute appears in more than one record type

   then - create a new record type consists of that attribute.

   - make this record type an owner for every record type contains this attribute.

   - delete this attribute from the record types containing it.
5. Create a system owned set for every N_i that is not a member in any set.

4.2 Software Development

The modified algorithm, mentioned in 4.1, is developed using the SADT included in Appendix D as one of the efficient software engineering techniques. Then the structure charts, included in Appendix A, are developed. The basic principles of software engineering are considered: minimum number of global variables, reasonable number of modules called by each module, and optimum number of callings for each module. The suitable data structure is chosen according to the software requirements. The algorithm of each module is designed to achieve its objective. The required software is developed using the PASCAL language, and tested and implemented under the UNIX operating system.

4.2.1 Input Parameters

The parameters of a relational schema, which is the input to the developed software, are: number of relations, number of attributes in the schema, maximum number of attributes in a key, and the maximum number of nonkey attributes in a relation. To develop the required software, the following constants are assumed at the declaration part of the software:

- maximum number of relations in the schema (n) is 10.
- maximum number of attributes in the schema (m) is 10.
- maximum number of attributes in a key (11) is 5.
- maximum number of nonkey attributes in a relation (12) is 9.

These values are suitable for some applications. But if any of the parameters of an application exceeds the maximum value, the value of the corresponding constant must be changed to suit the application otherwise an error message will be displayed and the software will stop the

execution. If the value of m or n is changed, the value of the constant mn, which is equal to m+n, must be changed also. The constant mn represents the maximum number of record types in the network schema.

4.2.2. Data Structure

The data structure used in the developed software is a combination of the static and dynamic types. This combination is chosen in such a way that it achieves two objectives: to be simple and easy to understand, and to use the necessary amount of space for each application. In addition to that, the software provides the user with the ability to change the values of the constants n, m, l1, l2, mn at its declaration part in order to suit the application. This ability minimizes the required space for an application.

The main part in the data structure is the type list1, illustrated in figure 4.1:

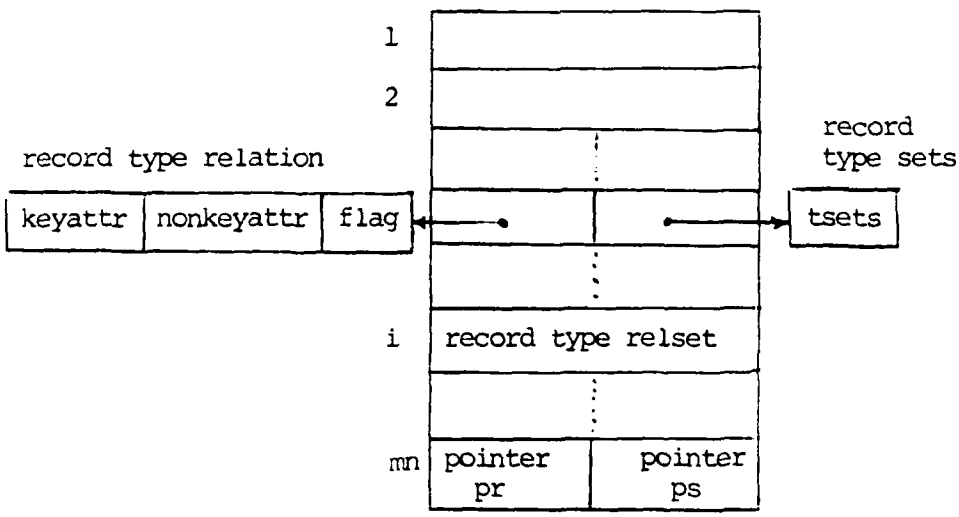


Figure 4.1. The Structure of Type list1

The data structure type listl is an array (1..mn) of type relset. Each record of type relset represents one record type in the network schema. Listl is a static data structure with respect to its length and mn is the maximum number of record types that can be created from a relational schema with parameters n and m. The record relset consists of two pointers: pr and ps which point to a record of type relation and sets respectively. The dynamic point in the data structure type listl is that all the pointers are initialized by nulls and one record of type relset is created only when a new record type in the network schema is created. This means that at any time the number of created records of type relset is equal to the number of record types in the network schema.

The record type relation consists of the following fields:

1. Keyattr, array (1..11) of characters, contains the key attributes of the corresponding record type.
2. Nonkeyattr, array(1..12) of characters, contains the other attributes in the record type.
3. Flag, character, contains a control value during the execution process.

The record type sets consists of one field, that is tsets, which is an array (1..mn) of integers. The value of an integer is used to identify if there is a set between two record types, or not. The integer rs(i).ps^.tsets(j) can be equal to 0, 1, or -1 according to the following:

- 0 if there is no set contains both record types i and j.
- 1 if there is a set and record i is its owner.
- -1 if there is a set and record i is its member.

The integer rs(i).ps^.tsets(i) is equal to 0 unless if the record i is a system owned record then it is equal to 2.

4.2.3 Input Format

The developed software is an interactive one. This means that the user has to prepare the required data in advance before using the software for implementation. The data will be input to the software through the keyboard when the user is asked to do. Slight modification can be done to the software in order to read the data from an input file. The required input data are:

1. The number of relations in the relational schema.
2. The number of attributes in the relational schema.
3. The attributes of the schema.
4. For each relation in the schema:
    i. the key attributes
    ii. the other attributes

For simplicity, the user has to assign one character symbol for each attribute in the relational schema and use it in the input of data.

4.2.4. Output Format

As mentioend before, in section 1.4, the output should be in an understandable form so that the user can easily draw the network schema and implement it. The output, that is the network schema, is designed to be as follows:

1. The number of record types.
2. The number of attributes.
3. The attributes.

4. For each record type in the schema:

   i. the record number

   ii. the key attributes

   iii. the other attributes

   iv. for each set includes the record type

      - the numbers of record types included in the set.

      - the owner of the set.

      - the member of the set.

   v. if the record type is a system owned record or not.

4.3 Software Testing

To test the developed software, the necessary checks and tests are done for each module within the software and for every part within each module. All the intermediate results are displayed during several runs. All the results are as expected and as supposed to be. After that, the tests are deleted from the software for the documentation purposes.

These are two main tests in the software to check the correctness of the input parameters m and n. Each of them has not to exceed its maximum value explained in 4.2.1. If any exceeds this value, an error message will be displayed and the program will stop the execution process.

To help the user avoiding this type of error, the software displays at the beginning of each run some paragraphs explaining its objectives, the maximum values of input parameters, and what the user can do to change these values. The complete documented software is included in Appendix B.

4.4 Implementation

The developed software is implemented using the same relational schema mentioned in Chapter 3 as an input to be translated into an equivalent network one. The resulting network schema, included in Appendix C, is exactly identical to the manual solution of Algorithm I in 3.1.2. Also, all the tests and intermediate results are coincident with the corresponding steps of the modified algorithm in 4.1. This implementation proves that the developed software achieves its objectives.

In addition to that, the software is implemented using different relational databases. The generated network databases are efficient, i.e., have the necessary and sufficient number of record types and sets.

V. Conclusions and Recommendations

5.1 Conclusions

In this study, an automated technique for translating a relational schema into an equivalent network one was developed. Several important conclusions were reached during the course of study. They are:

1. The multi-model DBMS eases many problems of the uni-model one. There are two approaches to achieve the multi-model DBMS: the mapping approach and the composite approaches.
2. The mapping approach is the suitable one for the conversion applications because it suggests that a schema for one model can be generated from the schema of the other model. The composite approach needs a special consideration during the schema design because it suggests that a schema for one model be embedded into schema of the other model.
3. The major step in creating a multi-model DBMS from a network one, is the translation of a relational schema into an equivalent network one. There are two algorithms that were developed for this purpose, but no one attempted to develop such an automated technique.
4. Algorithm I (10;14) provides the simpler methodology for the translation process. Also the generated network schema is simpler and more efficient than that of Algorithm II (12).
5. Developing such an automated technique is an important and necessary objective because it is a big step toward creating the multi-model DBMS, and it facilitates converting relational databases into network databases.

6. This technique was developed using PASCAL as one of the most common languages. It was designed to be user friendly, interactive, and to produce a network schema in an easy and understandable form.
7. The developed technique was tested and implemented using the same example used in the manual solutions in 3.1 and 3.2. The manual solution of algorithm I and the result of the developed technique were identical.
8. The automated technique was designed to produce an efficient network schema. There is no redundancy either in the record types or in the sets, and that keeps the number of record types and sets minimum.

5.2 Recommendations

The developed automated technique can be efficiently used in translating a relational schema into an equivalent network one. Based on the assumptions stated initially, the following recommendations are proposed for further study.

1. Developing an automated technique for converting a relational schema to its third normal form. This conversion is necessary before translating a relational schema into a network one.
2. Developing a technique for displaying the resulting network schema in a graphical form. This technique will help the user understanding the structure of the network schema easily and quickly.
3. Developing an automated technique for mapping the relational commands to equivalent network commands. This will provide the user with the ability to access the database using either relational or network commands.
4. Creating a multi-model DBMS from a network one using the results of the points mentioned before together with the automated technique developed in this thesis.
5. Results of points 1 and 2 can be used to modify the automated technique developed in this thesis effort so that it can handle translating a relational database, that is not in the third normal form, and displaying the results in a graphical form.

Bibliography

1. Teorey, Toby J. and James P. Fry. Design of Database Structure. Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

2. Martin, James. Principles of Database Management. Englewood Cliffs, New Jersey: Prentice-Hall, 1976.

3. Kroenke, David. Database Processing (Second Edition). Chicago: Science Research Associates, 1983.

4. Date, C.J. An Introduction to Database Systems, Volume I (Third Edition). California: Addison-Wesley Publishing Company, 1981.

5. Date, C.J. An Introduction to Database Systems, Volume II. California: Addison-Wesley Publishing Company, 1983.

6. Atre, S. Database: Structured Techniques for Design, Performance, and Management. New York: A Wiley-Interscience Publication, 1980.

7. Ullman, Jeffrey D. Principles of Database Systems (Second Edition). Maryland: Computer Science Press, 1982.

8. Cardenas, Alfonso F. Database Management Systems. Boston: Allyn and Bacon, 1979.

9. Nicely, Capt Debra J. The Operator Mapping Between Relational Algebra Operators and Codasyl Based Database Managed by a Codasyl DBMS. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, Dec 1983.

10. Kuck, Sharon McCure. A Design Methodology for a Universal Relation Scheme Implementation via Codasyl. PhD dissertation. University of Illinois, Urbana IL, 1982.

11. Larson, James A. "Bridging the Gap Between Network and Relational Database Management SYstems," IEEE Computer Vol. 16, No. 9: (82-92) (Sep 1983).

12. De, Prabuddha and others. "Toward an Optimal Design of a Network Database from Relational Descriptions," Operations Research, Vol. 26: 805-823 (Sep-Oct 1978).

13. Vassiliou, T. and Lochovsky F.H. "DBMS Transaction Translation," IEEE Proc. Compsac 80: 89-96 (Oct 1980).

14. Kuck, S.M. and Y. Sagiv "A Universal Relation Database System Implemented vis the Network Model," Proc. Symp. on Principles of Database Systems: 147-157 (1982).

15. Lien, Y. Edmund "On the Equivalence of Database Models," Journal of the Association for Computing Machinery, Vol. 29, No. 2: 333-362 (Apr 1982).

16. Travis, Capt Charles T. Interactive Automated System for Normalization of Relations. MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1983.

17. Brodie, Michael P., University of Maryland. "Research on the Translation and Standardization of Relational and Network Database Management Systems." Report to U.S. Army Research Office, March 1981.

18. Sibley, Edgar H., University of Maryland. "Problems in the Translation and Standardization of Relational and Network Type Database Management Systems." Report to U.S. Army Research Office, June 1977.

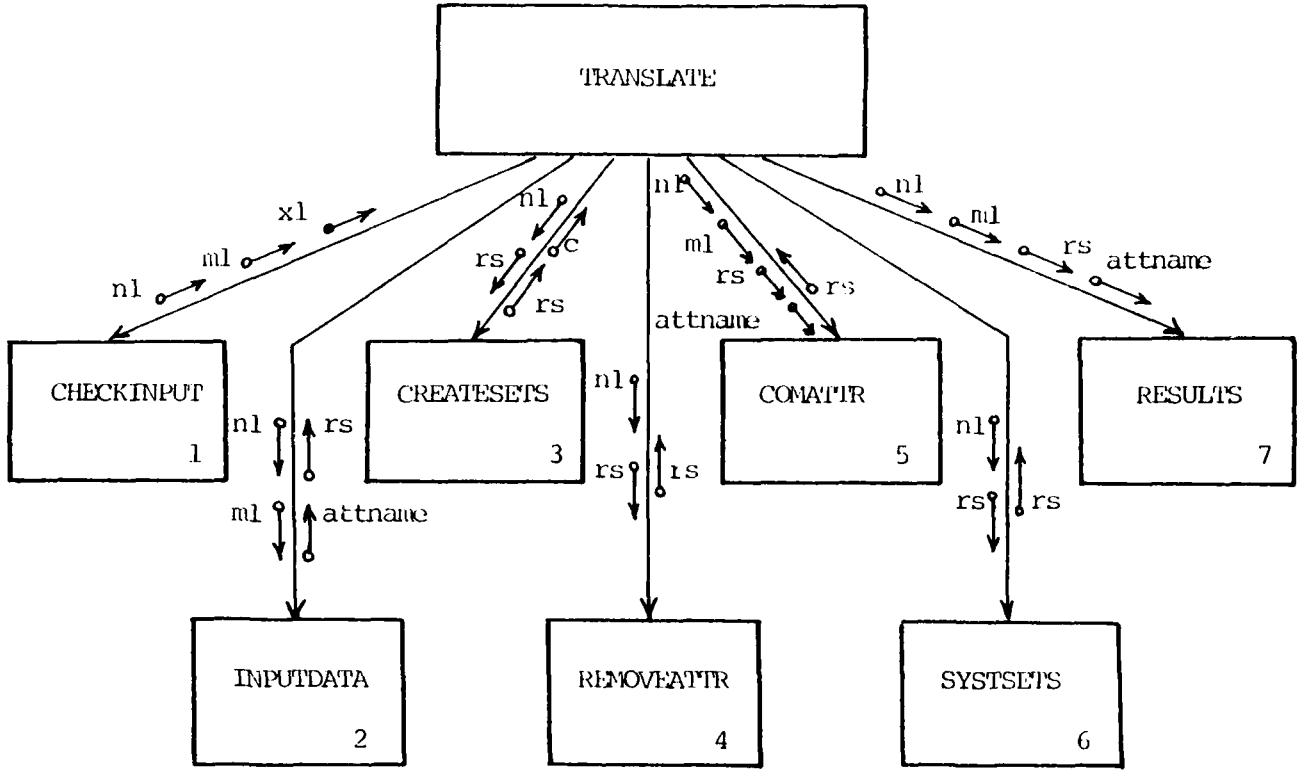
19. Zaniolo, C. "Design of Relational Views Over Network Schemas," Proc. ACM-SIGMOD Int. Conf. on Management of Data: 179-180 (1979).

20. Bernstein, P.A. "Synthesizing Third Normal Form Relations from Functional Dependencies," ACM Trans. on Database Systems, Vol. 1, No.4: 277-298 (Dec 1976).

21. Biskup, J. and others, "Synthesizing Independent Database Schemas," Proc. ACM-SIGMOD Int. Conf. on Management of Data: 143-151 (1979).

22. Peters, Lawrence J. Software Design: Methods and Techniques. New York: Yourdon Press, 1981.

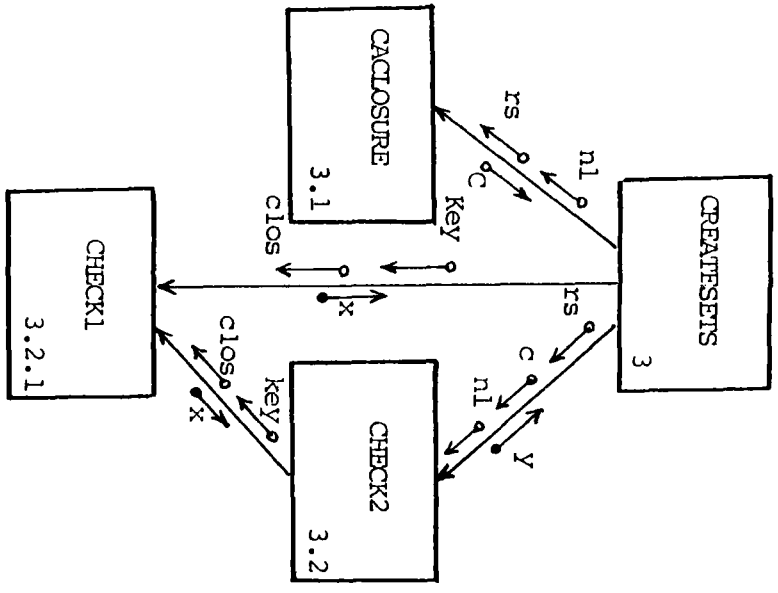
AUTHOR: Elsayed Yaser	DATE: 8/15/84	READER				
PROJECT: Translate	REV: / /	DATE				



Appendix A
Structure Charts

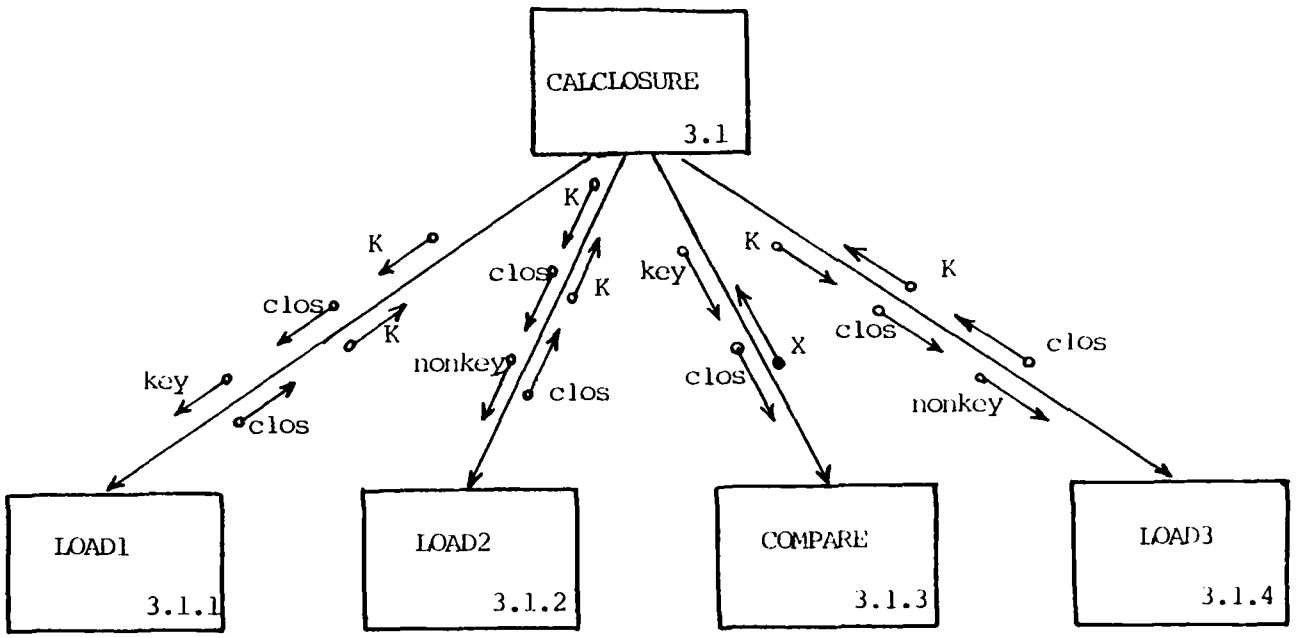
NOTE: A-0	TITLE: Translate A Relational Database Into an Equivalent Network One	NUMBER: G1

AUTHOR: Elsayed Yaser	DATE: 8/15/84	READER				
PROJECT: Translate	REV: / /	DATE				

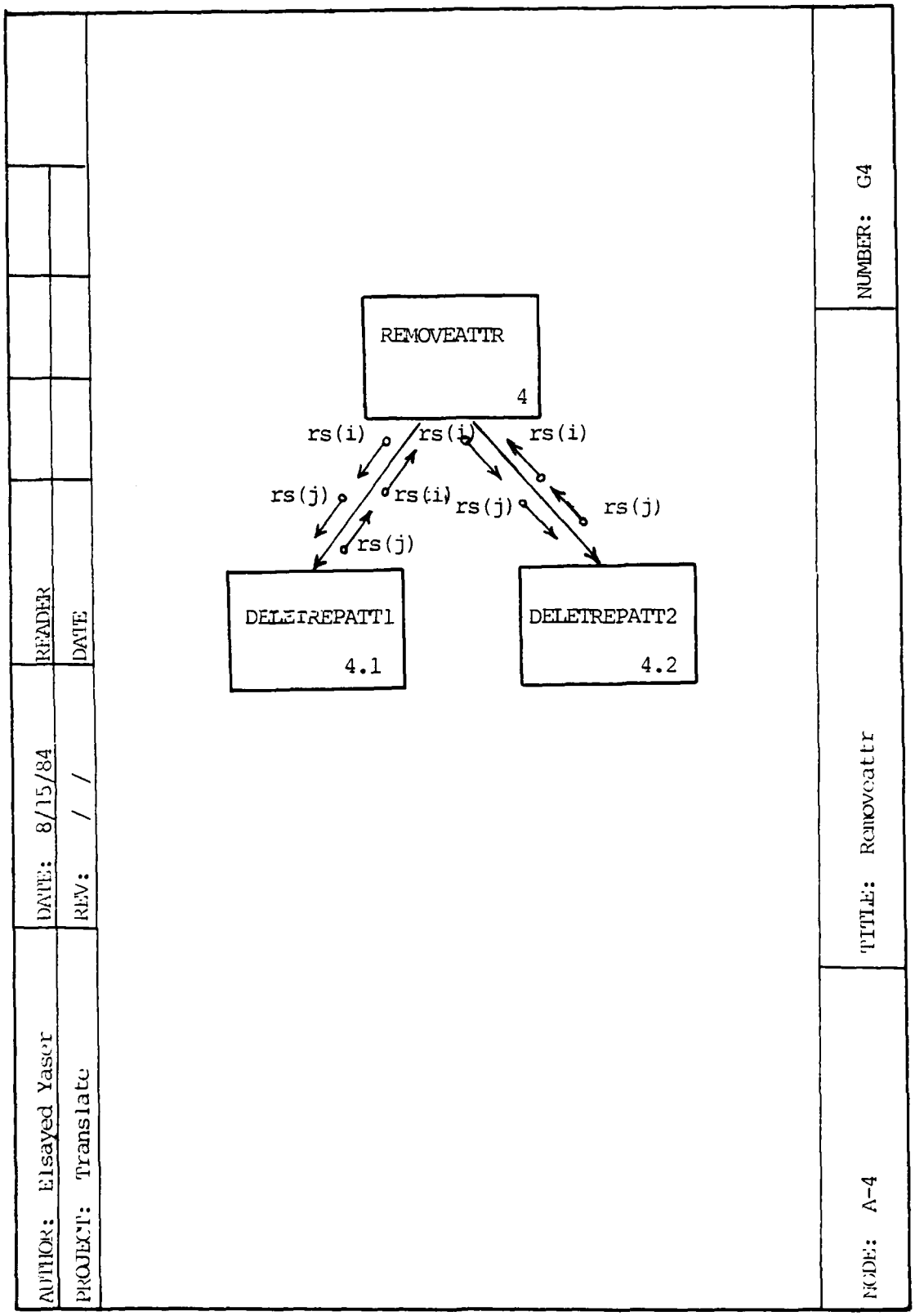


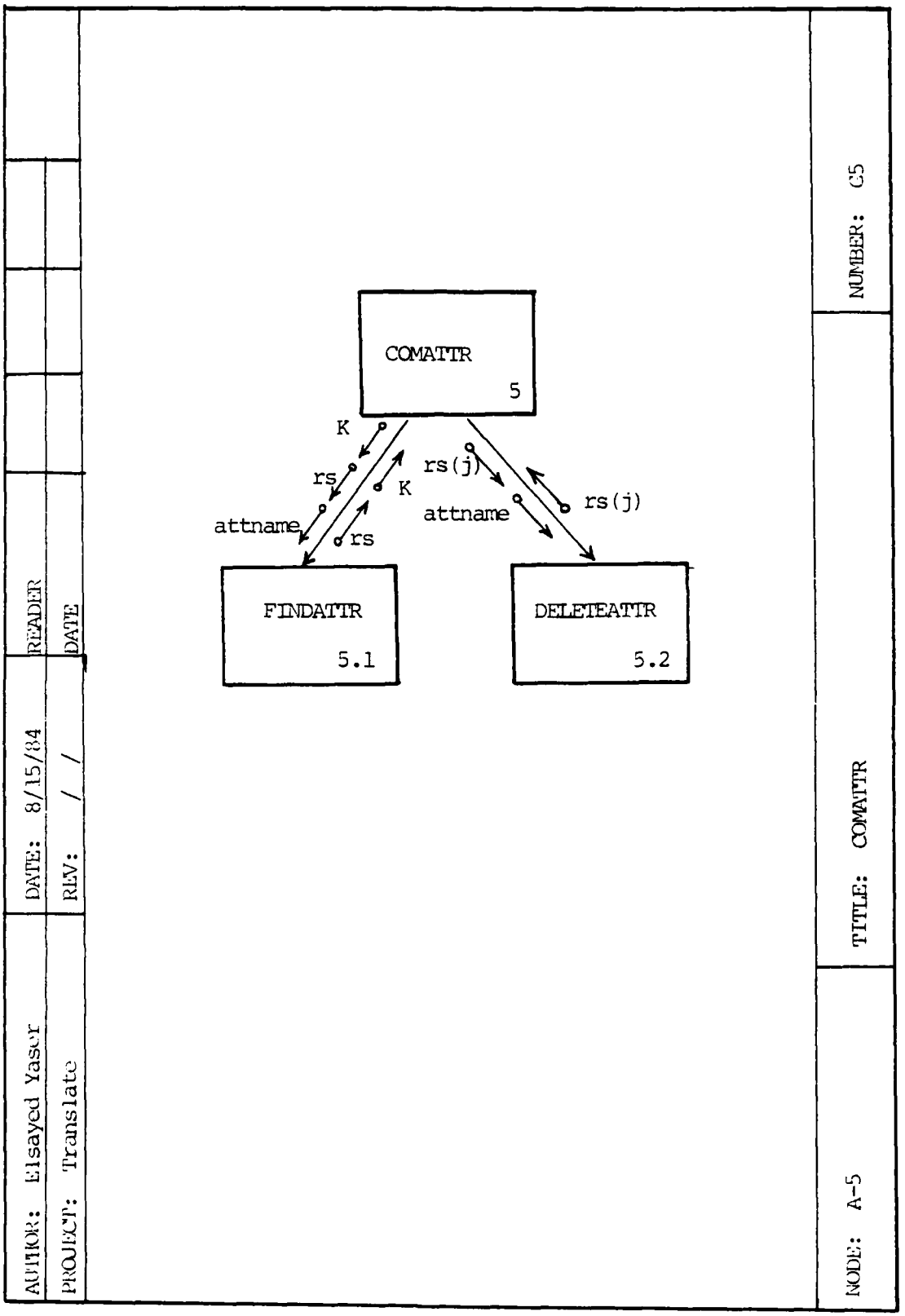
NOTE: A-3	TITLE: Createsets	NUMBER: G2

AUTHOR: Elsayed Yaser	DATE: 8/15/84	READER				
PROJECT: Translate	REV: / /	DATE				



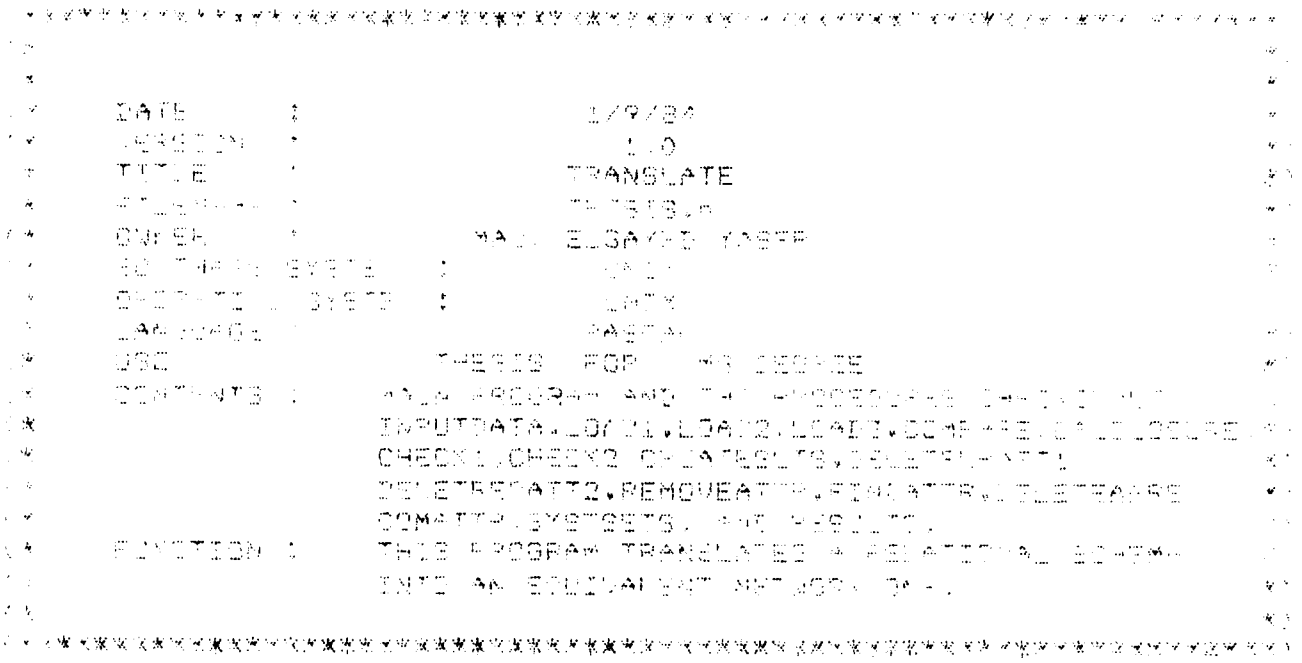
MODE: A-3.1	TITLE: Calclosure	NUMBER: G3


REMOVEATTR
4
rs(i)
rs(i)
rs(i)
rs(j)
rs(i)
rs(j)
rs(j)
rs(j)
DELETREPATT1
4.1
DELETREPATT2
4.2
AUTHOR: Elsayed Yasser
PROJECT: Translate
DATE: 8/15/84
REV: / /
READER
DATE
NUMBER: G4
TITLE: Removeattr
NODE: A-4


AUTHOR: Elsayed Yaser
PROJECT: Translate
DATE: 8/15/84
REV: / /
READER
DATE
NODE: A-5
TITLE: COMATTR
NUMBER: C5
COMATTR
5
K
rs
attname
K
rs
rs(j)
attname
rs(j)
FINDATTR
5.1
DELETEATTR
5.2

Appendix B

Developed Automated Technique



```
    program translate (input,output);

const

    n  = 10 ;          (* max. no. of relations is 10 *)
    m  = 11 ;          (* max. no. of attributes is 10 *)
    l1 =  6 ;          (* max. no. of key_attr. in a relation is 5 *)
    l2 = 10 ;          (* max. no. of nonkey_attr. in a rel. is 9 *)
    mn = 10 ;          (* mn = m n *)

type
    tpr = ^relation ;
    tps = ^sets     ;

   list4 = array [1..11] of char ;
   list5 = array [1..12] of char ;

   relation = record
                keyattr :  list4;
             nonkeyattr :  list5;
                   flag : char ;
      end; (* record *)

    sets =    record
              tsets    : array [1..mn] of integer ;
      end; (* record *)

   relset    = record
              pr  : tpr ;
              ps  : tps ;
      end; (* record *)

  list1 = array [1..mn] of relset ;
  list2 = array [1..m]  of char    ;
  list3 = record
            clos : list2 ;
     end;  (*record *)

  closure =  array [1..n] of list3 ;


var
   rs  : list1     ;
   c   : closure   ;
   attname     : list2    ;
   i,[illegible]   : integer ;
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*                                                               *)
(*     DATE      :           29/8/84                             *)
(*     VERSION   :             1.0                               *)
(*     NAME      :         Procedure checkinput                  *)
(*     MODULE NUMBER     :     1                                 *)
(*     FUNCTION :    Checks that the no of relations(nl) and no of *)
(*                   attributes(ml) of the input schema are within *)
(*                   the permissible range of the program.       *)
(*     INPUTS    :     Keyboard input                            *)
(*     OUTPUTS   :     nl,ml,and error messages                  *)
(*     GLOBAL VARIABLES USED      :     nl,ml                    *)
(*     GLOBAL VARIABLES CHANGED   :     nl,ml                    *)
(*     GLOBAL TABLES USED         :       -                      *)
(*     GLOBAL TABLES CHANGED      :       -                      *)
(*     FILES READ       :              -                        *)
(*     FILES WRITTEN    :              -                        *)
(*     MODULES CALLED   :              -                        *)
(*     CALLING MODULES :          Main program                  *)
(*     AUTHOR   :           Maj. Elegred Yaser Ali               *)
(*     HISTORY :                       -                       *)
(*                                                               *)
(*****************************************************************)
```

```
procedure checkinput (var n1,m1,k1:integer);

 var   i1:integer;

begin
  k1:=0;
  writeln('                WELCOME TO PROGRAM TRANSLATE              ');
  writeln;
  writeln;
  writeln('THE FUNCTION OF THIS PROGRAM IS TO TRANSLATE A RELATIONAL ');
  writeln('SCHEMA INTO AN EQUIVALENT NETWORK ONE , THE PROGRAM IS    ');
  writeln('DESIGNED TO HANDLE A RELATIONAL SCHEMA WITH THE FOLLOWING ');
  writeln('PARAMETERS :                                               ');
  writeln;
  writeln('MAX. NO. OF RELATIONS [n]  IS  10                          ');
  writeln('MAX. NO. OF ATTRIBUTES [m] IS  10                          ');
  writeln('MAX. NO. OF KEY_ATTR. [l1] IS   5                          ');
  writeln('MAX. NO. OF NONKEY_ATTR. [l2] IS  9                        ');
  writeln('MAX. NO. OF RECORD TYPES IN THE GENERATED NETWORK SCHEMA  ');
  writeln('[mn] IS 20 WHICH IS EQUAL TO m+n .                         ');
  for i1:=1 to 4 do
    writeln;
  writeln('IF ANY OF YOUR SCHEMA PARAMETERS EXCEEDS THESE MAX. VALUES,');
  writeln('PLEASE MODIFY THE CORRESPONDING CONSTANTS AT THE DECLARATION');
  writeln('PART OF THE PROGRAM TO SUIT YOUR NEEDS .                   ');
  writeln;
  writeln;
  writeln('                      THANK YOU                            ');
  for i1:=1 to 5 do
    writeln;
  writeln('      Enter the no of relations in the schema ');
  read(n1);
  if (n1>n) then
    begin
      k1:=1;
      writeln('ERROR:this no is greater than the permissible.');
      writeln('Modify the constants n, mn at the declaration');
      writeln('part of the program to suit your needs.');
    end
             else
    begin
      writeln('   Enter the no of attributes in the schema ');
      read(m1);
      if (m1 > (mn-1)) then
        begin
          k1:=1;
          writeln('ERROR:this no is greater than the permissible.');
          writeln('Modify the constants m,mn at the declaration');
          writeln('part of the program to suit your needs.');
        end; (* if *)
      end; (* if *)
    end; (* procedure checkinput *)
```

```
(*****************************************************************************)
(*                                                                           *)
(*                                                                           *)
(*    DATE       :              29/8/84                                      *)
(*    VERSION    :                 1.0                                       *)
(*    NAME       :          Procedure inputdata                              *)
(*    MODULE NUMBER     :          2                                         *)
(*    FUNCTION   :      Inputs the relational scheme from the user,          *)
(*                      and creates the data structure.                      *)
(*    INPUTS     :             n1,m1                                         *)
(*    OUTPUTS    :          rs,attname                                       *)
(*    GLOBAL VARIABLES USED       :       n1,m1,rs,attname                   *)
(*    GLOBAL VARIABLES CHANGED    :       rs,attname                         *)
(*    GLOBAL TABLES USED          :              -                           *)
(*    GLOBAL TABLES CHANGED       :              -                           *)
(*    FILES READ        :                        -                           *)
(*    FILES WRITTEN     :                        -                           *)
(*    MODULES CALLED    :                        -                           *)
(*    CALLING MODULES   :               main program                         *)
(*    AUTHOR   :                    Maj.Elsayed Hasan Aly                    *)
(*    HISTORY  :                                 -                           *)
(*                                                                           *)
(*****************************************************************************)
```

```
procedure inputdata ( var rs : list1 ;var attname : list2; al,nl : integer );

 var     i,j : integer ;

 begin
  reset(input);
  writeln ('Enter the attribute symbols, each is one char. only .');
  writeln ('Symbols must be in one line without separating spaces ');
  for i:=1 to al do
     read(attname[i]);
  attname[al+1] := '*';
  for i:=1 to nl do
  begin
    new (rs[i].pr);
    new (rs[i].ps);
    with rs[i].pr^ do
    begin
      reset(input);
      writeln('Enter the Key_attribute symbols of relation no ',i );
      j := 1 ;
      while not eoln do
      begin
        read (Keyattr[j]);
        j := j+1 ;
      end;
      Keyattr[j] := '*' ;
      reset(input);
      writeln('Enter the nonKey_attribute symbols of the relation ');
      j:= 1 ;
      while not eoln do
      begin
        read (nonKeyattr[j]);
        j := j+1 ;
      end ;
      nonKeyattr[j] := '*';
      flag := '0' ;
    end ; (*  with *)
    with rs[i].ps^ do
      for j:=1 to an do
         tsets[j] := 0 ;
  end ;  (* for *)
end ;  (* procedure inputdata *)
```

```
(*****************************************************************************
(*                                                                           *
(*                                                                           *
(*     DATE      :                29/9/84                                    *
(*     VERSION   :                  1.0                                      *
(*     NAME      :            Procedure load1                                *
(*     MODULE NUMBER     :       3.1.1                                      *
(*     FUNCTION :     Loads the key attributes of a relation in             *
(*                    its closure.                                           *
(*     INPUTS    :      rs[i].pr^.Keyattr,cl[i].clos.r                       *
(*     OUTPUTS   :      cl[i].clos , K                                       *
(*     GLOBAL VARIABLES USED      :     rs,i,K                              *
(*     GLOBAL VARIABLES CHANGED   :      c,K                                *
(*     GLOBAL TABLES USED         :        _                                 *
(*     GLOBAL TABLES CHANGED      :        _                                 *
(*     FILES READ        :                 _                                 *
(*     FILES WRITTEN     :                 _                                 *
(*     MODULES CALLED    :                 _                                 *
(*     CALLING MODULES :        Procedure closure                            *
(*     AUTHOR    :              Mr. Elsayed Yasser Aly                       *
(*     HISTORY :                          _                                  *
(*                                                                           *
(*****************************************************************************
```

```
procedure local (var x : list1 ; var y : list2 ; var z : integer);

var
    n   : integer ;

begin
  z := 1 ;
  while (y <> nil) do
  begin
    x^[z] := y^.val ;
    k     := k + 1 ;
    n     := n + 1 ;
  end;  (* while *)
end ; (* procedure local *)
```

```
(*                                                                    *)
(*                                                                    *)
(*     DATE      :              29/6/84                               *)
(*     VERSION   :                1.0                                *)
(*     NAME      :           Procedure load2                         *)
(*     MODULE NUMBER     :      3.1.2                                *)
(*     FUNCTION  :     Loads the non-ev attributes of a relation      *)
(*                     in its closure.                                *)
(*     INPUTS    :     rcIls.prA.conkeyattr,cIlI.clos.K             *)
(*     OUTPUTS   :     cIlI,clos.K                                    *)
(*     GLOBAL VARIABLES USED      :      rc,c.K                       *)
(*     GLOBAL VARIABLES CHANGED   :      c.K                          *)
(*     GLOBAL TABLES USED         :        -                          *)
(*     GLOBAL TABLES CHANGED      :        -                          *)
(*     FILES READ       :                 -                          *)
(*     FILES WRITTEN    :                 -                          *)
(*     MODULES CALLED   :                 -                          *)
(*     CALLING MODULES  :       Procedure calcclosure                *)
(*     AUTHOR    :              Maj. Elsayed Yasser A.               *)
(*     HISTORY   :                       -                            *)
(*                                                                    *)
(**********************************************************************)
```

```
procedure load2 ( var sv : list2 ; var nx : listE ) var x : integer
var
    z : integer ;
begin
  z := 1 ;
  while sv[z] <> (*) do
  begin
    b[z] := sv[z] ;
    x      := x + 1 ;
    z      := z + 1 ;
  end; (* while *)
end; (* procedure load2 *)
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :                30/8/84                        *)
(*     VERSION   :                  1.0                          *)
(*     NAME      :             Procedure load3                   *)
(*     MODULE NUMBER     :        3.1.4                          *)
(*     FUNCTION :      Loads the nonkey attributes of a relation  *)
(*                     in the closure of another one. Before load *)
(*                     ing any attribute, the procedure checks that *)
(*                     it is not already exist in the closure.   *)
(*     INPUTS    :     rs[i].pr^.nonkeyattr,cli,rlec,k           *)
(*     OUTPUTS   :     cli.clos.k,r                              *)
(*     GLOBAL VARIABLES USED      :       rs,c,k,y               *)
(*     GLOBAL VARIABLES CHANGED   :       c,k,r                  *)
(*     GLOBAL TABLES USED         :          -                   *)
(*     GLOBAL TABLES CHANGED      :          -                   *)
(*     FILES READ       :                 -                     *)
(*     FILES WRITTEN    :                 -                     *)
(*     MODULES CALLED   :                 -                     *)
(*     CALLING MODULES :          Procedure calclo               *)
(*     CALLING MODULES :          Procedure calclosure           *)
(*     AUTHOR    :               Maj.Elsayed Yaser Aly          *)
(*     HISTORY :                          -                      *)
(*                                                               *)
(*****************************************************************)
```

```
procedure load3 ( var tv:list5; var ty:list2 ; var k,y:integer);

  var   k1,y1,i1,j1 : integer ;

begin
  k1 := k -1 ;
  i1 := 1    ;
  while( tv[i1] <>'*') do
  begin
    j1 := i1;
    y1 := 1;
    while (( j1 <= k1) and( y1=1) )  do
      if( tv[i1] =ty[j1]) then y1:= 0
                          else j1:=j1+1;
    if (( j1>k1) and( y1=1 )) then
      begin
        y := 1 ;
        ty[k] := tv[i1];
        k     := k + 1 ;
      end;
    i1 := i1 + 1 ;
  end ; (* while *)
end ; (* procedure load3 *)
```

```
(*                                                                    *)
(*                                                                    *)
(*     DATE      :              30/8/84                               *)
(*     VERSION   :                1.0                                *)
(*     NAME      :         Procedure compare                         *)
(*     MODULE NUMBER     :      3.1.3                                *)
(*     FUNCTION  :     Checks if the key attributes of a relation    *)
(*                     belongs to the closure of another relation    *)
(*     INPUTS    :     rel1.pr^.keyattr,cl1,clos,k                   *)
(*     OUTPUTS   :     x                                             *)
(*     GLOBAL VARIABLES USED      :       rs,c,k,x                   *)
(*     GLOBAL VARIABLES CHANGED   :          x                       *)
(*     GLOBAL TABLES USED         :          -                       *)
(*     GLOBAL TABLES CHANGED      :          -                       *)
(*     FILES READ       :                    -                       *)
(*     FILES WRITTEN    :                    -                       *)
(*     MODULES CALLED   :                    -                       *)
(*     CALLING MODULES  :          Procedure calclosure              *)
(*     AUTHOR    :                 Maj.Elsayed Yaser Aly             *)
(*     HISTORY   :                          -                        *)
(*                                                                    *)
(**********************************************************************)
```

```
procedure compare ( var x : list1 ; var y : list2 ; var k,x : integer );

var
   k1,y1,i1,j1 : integer ;

begin
  k1 := k-1;
  y1 := 0;
  i1 := 1;
  j1 := 1;

  while ((u[i1]<>'*') and (y1=0)) do
  begin
    y1 := 1;
    j1 := 1;
    while (( j1 <= k1) and( y1= 1)) do
       if ( u[i1] = v[j1] )
         then
           begin
             i1 := i1 + 1;
             y1 := 0 ;
           end
         else  j1 := j1 + 1;
    end ; (* while *)
    if( ( u[i1]='*') and( y1= 0 ))  then x := 0 ;
end ; (* procedure compare *)
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :            1/9/84                             *)
(*     VERSION   :             1.0                               *)
(*     NAME      :    Procedure calclosure                       *)
(*     MODULE NUMBER     :    3.1                                *)
(*     FUNCTION  :    Calculates the closure of each relation    *)
(*     INPUTS    :              rs,n1                            *)
(*     OUTPUTS   :               c                               *)
(*     GLOBAL VARIABLES USED     :    rs,c,n1                    *)
(*     GLOBAL VARIABLES CHANGED  :       c                       *)
(*     GLOBAL TABLES USED        :       _                       *)
(*     GLOBAL TABLES CHANGED     :       _                       *)
(*     FILES READ       :                _                       *)
(*     FILES WRITTEN    :                _                       *)
(*     MODULES CALLED   :   Procedures local1,local2,local7,comm1 *)
(*     CALLING MODULES  :          Procedure createsets           *)
(*     AUTHOR    :                 Maj,klsayed Yaser Aly          *)
(*     HISTORY   :                       _                       *)
(*                                                               *)
(*****************************************************************)
```

```
procedure calclosure( var c:closure; rellist:nilinteger );

var  i,x,y : integer ;

begin
  for i:=1 to n1 do
  begin
    k := 1 ;
    load1(c[i].clos,rel[i].pr^.keyattr,k);
    load2(c[i].clos,rel[i].pr^.nonkeyattr,x);
    y:=1;
    while (y=1) do
      begin
        y:=0;
        for j:=1 to n1 do
          if i<>j then
            begin
              compare(rel[j].pr^.keyattr,c[i].clos,k,x);
              if x=0 then
                load3(rel[j].pr^.nonkeyattr,c[i].clos,k,y);
            end; (* if *)
      end; (* while *)
    c[i].clos[k] := '*';
  end ; (* for *)
end ; (* procedure calclosure *)
```

```
(*                                                                          *)
(*                                                                          *)
(*     DATE      :              2/9/84                                      *)
(*     VERSION   :               1.0                                        *)
(*     NAME      :        Procedure check1                                  *)
(*     MODULE NUMBER     :    3.2.1                                         *)
(*     FUNCTION :       Checks that a relation belongs to the              *)
(*                      closure of another one.                            *)
(*     INPUTS    :       rs[i].pr^.keyattr,cli.clsc                        *)
(*     OUTPUTS   :       x                                                  *)
(*     GLOBAL VARIABLES USED      :      rs.c,x                            *)
(*     GLOBAL VARIABLES CHANGED   :         x                              *)
(*     GLOBAL TABLES USED         :         -                              *)
(*     GLOBAL TABLES CHANGED      :         -                              *)
(*     FILES READ       :                  -                              *)
(*     FILES WRITTEN    :                  -                              *)
(*     MODULES CALLED   :                  -                              *)
(*     CALLING MODULES :     Procedures check2.createsets                  *)
(*     AUTHOR   :                 Maj.Elsayed Yaser Alsets                 *)
(*     AUTHOR   :                   Maj.Elsayed Yaser Ali                  *)
(*     HISTORY :                         -                                 *)
(*                                                                          *)
(****************************************************************************)
```

```
procedure check1 ( var a:list1; var b:list2; var x:integer);

var i1,j1,y1 : integer ;

begin
  i1:=1;
  x :=1;
  y1:=0;
  while ((a[i1]<>'*') and (y1=0)) do
    begin
      y1:=1;
      j1:=1;
      while ((b[,j1]<>'*') and (y1=1)) do
        if (a[i1]=b[,j1])
          then
            begin
              y1:=0;
              i1:=i1+1;
            end
          else  j1:=j1+1;
    end;  (* while *)
  if (( a[i1]='*') and (y1=0)) then  x:=0;
end; (* procedure check1 *)
```

```
(*****************************************************************************)
(*                                                                           *)
(*                                                                           *)
(*     DATE      :                2/9/84                                     *)
(*     VERSION   :                 1.0                                       *)
(*     NAME      :         Procedure check2                                  *)
(*     MODULE NUMBER     :         3.3                                       *)
(*     FUNCTION :  Checks that there is no Rk such that it belongs           *)
(*                 to Rk+1 and Rk+1 belongs to Rk for a given i,j .          *)
(*     INPUTS    :              rs,c,i,j,n1                                  *)
(*     OUTPUTS   :                  y                                        *)
(*     GLOBAL VARIABLES USED      :       rs,c,i,j,n1,y                      *)
(*     GLOBAL VARIABLES CHANGED   :            y                             *)
(*     GLOBAL TABLES USED         :            _                             *)
(*     GLOBAL TABLES CHANGED      :            _                             *)
(*     FILES READ       :                      _                             *)
(*     FILES WRITTEN    :                      _                             *)
(*     MODULES CALLED   :            Procedure check1                       *)
(*     CALLING MODULES  :           Procedure creatcase                     *)
(*     AUTHOR    :                  Maj.Elsayed Yaser Aly                   *)
(*     HISTORY :                            _                                *)
(*                                                                           *)
(*****************************************************************************)
```

```
procedure check2 ( clclosure; rs:list1;i,j,n1:integer;var x:integer);

 var k :integer ;

 begin
   k:=1;
   y:=0;
   while ((y=0)and(k<=n1)) do
     begin
       if((k<>i)and(k<>j)) then
         begin
           check1(rs[i],nr^.keyattr,cl[i].clos,x );
           if (x=0) then
             begin
               check1(rs[k],nr^.keyattr,cl[j].clos,x);
               if (x=0) then y:= 1;
             end;  (* if *)
         end;  (* if *)
     k:=k+1;
     end;  (* while *)
  end;  (* procedure check2 *)
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :              3/9/84                           *)
(*     VERSION   :               1.0                             *)
(*     NAME      :        Procedure createsets                   *)
(*     MODULE NUMBER    :         3                              *)
(*     FUNCTION :      Creates the sets between the record types.*)
(*     INPUTS    :              rs,rl                            *)
(*     OUTPUTS   :              rs,c                             *)
(*     GLOBAL VARIABLES USED      :        rs,c,rl               *)
(*     GLOBAL VARIABLES CHANGED   :         rs,c                 *)
(*     GLOBAL TABLES USED         :           -                  *)
(*     GLOBAL TABLES CHANGED      :           -                  *)
(*     FILES READ       :                    -                  *)
(*     FILES WRITTEN    :                    -                  *)
(*     MODULES CALLED   :    Procedures calcl&sure,check1,check2 *)
(*     CALLING MODULES :              Main program               *)
(*     AUTHOR   :                  Maj.Elsayed Yaser Ali         *)
(*     HISTORY :                       -                         *)
(*                                                               *)
(*****************************************************************)
```

```
procedure createsets ( var c:closure; var rs:list; n1 : integer );

var  x,y : integer ;

begin
   calclosure(c,rs,n1);
   for i:=1 to n1 do
      for j:=1 to n1 do
         if (i<j) then
            begin
               check1(rs[i],rs^.[illegible]);
               if (x=0) then
                  begin
                     check2(c,rs,i,j,n1,y);
                     if (y=0) then
                        begin
                           rs[i].ps^.[illegible][j] := 1;
                           rs[j].ps^.tsets[i] :=-1;
                        end; (* if *)
                  end; (* if *)
            end; (* if *)
   end; (* procedure createsets *)
```

```
/*****************************************************************/
/*                                                               */
/*                                                               */
/*     DATE      :              4/9/84                           */
/*     VERSION   :                1.0                           */
/*     NAME      :       Procedure deletmbatt1                  */
/*     MODULE NUMBER     :        4.1                           */
/*     FUNCTION :      For each set, the procedure deletes from the */
/*                     member record type all attributes that belong */
/*                     to key attributes of the owner record type.  */
/*     INPUTS    :             rs[i].pr^,rs[j].nt^              */
/*     OUTPUTS   :             rs[i].pr^,rs[j].pr^              */
/*     GLOBAL VARIABLES USED     :              rs               */
/*     GLOBAL VARIABLES CHANGED  :              rs               */
/*     GLOBAL TABLES USED        :              -                */
/*     GLOBAL TABLES CHANGED     :              -                */
/*     FILES READ        :                      -                */
/*     FILES WRITTEN     :                      -                */
/*     MODULES CALLED    :                      -                */
/*     CALLING MODULES   :          Procedure removeattr         */
/*     AUTHOR    :                  Maj.Gilmored Yaser Ali       */
/*     HISTORY   :                              -                */
/*                                                               */
/*****************************************************************/
```

```
procedure deletrepeatt ( var u,v : relation);

var i1,j1,y1 : integer;

begin
  i1 := 1 ;
  while ( u.keyattr[i1] <> '*') do
    if ( u.keyattr[i1] = ' ')
      then   i1 := i1 + 1
      else  begin
              j1 := 1;
              y1 := 1;
              while ((v.keyattr[j1] <> '*')and(y1=1)) do
                if (u.keyattr[i1]=v.keyattr[j1])
                  then    begin
                            y1 := 0 ;
                            u.keyattr[i1] := ' ';
                            i1 := i1 + 1;
                          end
                  else    j1 := j1 + 1 ;
                if ( y1 = 1 )  then
                  begin
                    j1 := 1;
                    while ((v.nonkeyattr[j1]<>'*')and(y1=1)) do
                      if (u.keyattr[i1] = v.nonkeyattr[j1])
                        then    begin
                                  y1 := 0;
                                  u.keyattr[i1] := ' ';
                                end
                        else    j1 := j1 + 1;
                    i1 := i1 + 1;
                  end; (* if *)
            end;  (* if *)
    end;  (* procedure deletrepatt *)
```

```
(*****************************************************************************)
(*                                                                           *)
(*     DATE      :                4/9/84                                     *)
(*     VERSION   :                 1.0                                       *)
(*     NAME      :        Procedure deletrepatt2                             *)
(*     MODULE NUMBER     :         4.2                                       *)
(*     FUNCTION :      For each set, the procedure deletes from the         *)
(*                     member record type any attributes that belong        *)
(*                     to nonkey attributes of the owner record type.       *)
(*     INPUTS    :                rs[i].ps^.rs[i].ps^                       *)
(*     OUTPUTS   :                rs[i].os^.rs[i].os^                       *)
(*     GLOBAL VARIABLES USED      :             rs                           *)
(*     GLOBAL VARIABLES CHANGED   :             rs                           *)
(*     GLOBAL TABLES USED         :              -                           *)
(*     GLOBAL TABLES CHANGED      :              -                           *)
(*     FILES READ       :                        -                           *)
(*     FILES WRITTEN    :                        -                           *)
(*     MODULES CALLED   :                        -                           *)
(*     CALLING MODULES :           Procedure removeattr                      *)
(*     AUTHOR   :                 Maj.Elsayed Yaser Aly                      *)
(*     HISTORY :                          -                                  *)
(*                                                                           *)
(*****************************************************************************)
```

```
procedure deletrenatt2 ( var u,v : relation);

var i1,j1,y1 : integer;

begin
  i1 := 1 ;
  while ( u.nonkeyattr[i1] <> '*') do
    if ( u.nonkeyattr[i1] = ' ')
      then   i1 := i1 + 1
      else  begin
              j1 := 1;
              y1 := 1;
              while ((v.keyattr[j1] <> '*')and(y1=1)) do
                if (u.nonkeyattr[i1]=v.keyattr[j1])
                  then    begin
                            y1 := 0 ;
                            u.nonkeyattr[i1] := ' ';
                            i1 := i1 + 1;
                          end
                  else    j1 := j1 + 1 ;
              if ( y1 = 1 )  then
                begin
                  j1 := 1;
                  while ((v.nonkeyattr[j1]<>'*')and(y1=1)) do
                    if (u.nonkeyattr[i1] = v.nonkeyattr[j1])
                      then    begin
                                y1 := 0;
                                u.nonkeyattr[i1] := ' ';
                              end
                      else    j1 := j1 + 1;
                  i1 := i1 + 1;
                end; (* if *)
            end;   (* if *)
      end;   (* procedure deletrenatt2 *)
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :                4/9/84                         *)
(*     VERSION   :                  1.0                          *)
(*     NAME      :        Procedure removestr                    *)
(*     MODULE NUMBER     :          4                            *)
(*     FUNCTION :       For each record type ,the procedure removes  *)
(*                      any attribute that appears in its instance.  *)
(*     INPUTS    :                rs.n1                          *)
(*     OUTPUTS   :                rs                             *)
(*     GLOBAL VARIABLES USED      :          rs.n1                *)
(*     GLOBAL VARIABLES CHANGED   :           rs                  *)
(*     GLOBAL TABLES USED         :           _                   *)
(*     GLOBAL TABLES CHANGED      :           _                   *)
(*     FILES READ       :                    _                   *)
(*     FILES WRITTEN    :                    _                   *)
(*     MODULES CALLED   :    Procedures deletrepatt1,deletrepatt2  *)
(*     CALLING MODULES  :              Main program               *)
(*     AUTHOR    :                Maj.Elsayed Yaser Aly           *)
(*     HISTORY :                          _                      *)
(*                                                               *)
(*****************************************************************)
```

```
procedure removeattr ( var rs:listt; n1:integer );

var  i,j :integer ;

begin
  for i:=1 to n1 do
    for j:=1 to n1 do
      if (rs[i].ps^.tsets[j] =-1)
        then  begin
                deletrepatt1(rs[i].pr^,rs[j].pr^);
                deletrepatt2(rs[i].pr^,rs[j].pr^);
              end;  (* if *)
end;   (* procedure removeattr *)
```

```
(*                                                                          *)
(*                                                                          *)
(*     DATE      :                    6/9/84                               *)
(*     VERSION   :                     1.0                                 *)
(*     NAME      :            Procedure findattr                           *)
(*     MODULE NUMBER     :             5.1                                 *)
(*     FUNCTION :     Calculates how many times an attribute is            *)
(*                    repeated within the relational schema.               *)
(*     INPUTS    :               attname[i],rs,ni                          *)
(*     OUTPUTS   :               K                                         *)
(*     GLOBAL VARIABLES USED      :      attname,rs,ni,k                   *)
(*     GLOBAL VARIABLES CHANGED   :               k                       *)
(*     GLOBAL TABLES USED         :               -                       *)
(*     GLOBAL TABLES CHANGED      :               -                       *)
(*     FILES READ       :                         -                       *)
(*     FILES WRITTEN    :                         -                       *)
(*     MODULES CALLED   :                         -                       *)
(*     CALLING MODULES :             Procedure removeattr                 *)
(*     AUTHOR   :                    Maj.Elsayed Yasser ALY               *)
(*     HISTORY :                              -                           *)
(*                                                                          *)
(****************************************************************************)
```

```
procedure findattr ( var att:char; var rs:list1; n2:integer;
                     var K:integer);


var  i1,j1,y1 : integer;


begin
  for i1:=1 to n2 do
    begin
      y1:=1;
      with rs[i1].pr^  do
        begin
          j1:=1;
          while ((Keyattr[j1]<>'*')and(y1=1)) do
            if (Keyattr[j1]=att)
             then
               begin
                 K:=K+1;
                 y1:=0;
                 flag :='1';
               end
             else    j1:=j1+1;
          if (y1=1)
           then
             begin
               j1:=1;
               while((nonKeyattr[j1]<>'*')and(y1=1)) do
                if ( nonKeyattr[j1]=att )
                 then
                   begin
                     K:=K+1;
                     y1:=0;
                     flag :='1';
                   end
                  else j1:=j1+1;
             end;  (* if *)
        end;  (* with *)
      end;  (* for *)
  end; (* procedure findattr *)
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :                  6/9/84                       *)
(*     VERSION   :                   1.0                         *)
(*     NAME      :           Procedure deleteattr                *)
(*     MODULE NUMBER     :          5.2                          *)
(*     FUNCTION :   Deletes a  common attribute from a record type. *)
(*     INPUTS    :          attname,rs[i].pr^                    *)
(*     OUTPUTS   :             rs[i].pr^                         *)
(*     GLOBAL VARIABLES USED      :          rs,attname          *)
(*     GLOBAL VARIABLES CHANGED   :             rs               *)
(*     GLOBAL TABLES USED         :                              *)
(*     GLOBAL TABLES CHANGED      :                              *)
(*     FILES READ        :                                       *)
(*     FILES WRITTEN     :                                       *)
(*     MODULES CALLED    :                                       *)
(*     CALLING MODULES :              Procedure removeattr       *)
(*     AUTHOR    :                   Maj.Elsayed Nasr Aly        *)
(*     HISTORY :                                                 *)
(*                                                               *)
(*****************************************************************)
```

```
procedure deleteattr ( var att:char; var u:relation);

 var i1,y1 :integer;

 begin
   i1:=1;
   y1:=1;
   while ((u.keyattr[i1]<>'*')and(y1=1)) do
     if ( u.keyattr[i1]=att )
       then
        begin
          u.keyattr[i1] := ' ';
          y1 :=0;
        end
       else  i1:=i1+1;
   if (y1=1)  then
     begin
       i1:=1;
       while ((u.nonkeyattr[i1]<>'*')and(y1=1)) do
         if (u.nonkeyattr[i1]=att)
          then
            begin
              u.nonkeyattr[i1]:=' ';
              y1:=0;
            end
          else i1:=i1+1;
     end;      (* if *)
     u.flag :='0';
   end;    (* procedure deleteattr *)
```

```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :                  6/9/84                       *)
(*     VERSION   :                    1.0                        *)
(*     NAME      :              Procedure comattr                *)
(*     MODULE NUMBER     :            5                          *)
(*     FUNCTION :     Creates a new record type for each attribute *)
(*                    that appears more than once in the network  *)
(*                    schema ,and creates the appropriate sets for *)
(*                    that record.                               *)
(*     INPUTS    :           rs,attname,n1,m1                     *)
(*     OUTPUTS   :           rs,n1                                *)
(*     GLOBAL VARIABLES USED     :        rs,attname,n1,m1        *)
(*     GLOBAL VARIABLES CHANGED  :            rs,n1              *)
(*     GLOBAL TABLES USED        :              _                *)
(*     GLOBAL TABLES CHANGED     :              _                *)
(*     FILES READ       :                     _                 *)
(*     FILES WRITTEN    :                     _                 *)
(*     MODULES CALLED   :        Procedures findattr,deleteattr  *)
(*     CALLING MODULES :                 Main program            *)
(*     AUTHOR  :                    Maj.Elsayed Yaser Ali        *)
(*     AUTHOR  :                    Maj.Elsayed Yaser Ali        *)
(*     HISTORY :                          _                      *)
(*                                                               *)
(*****************************************************************)
```

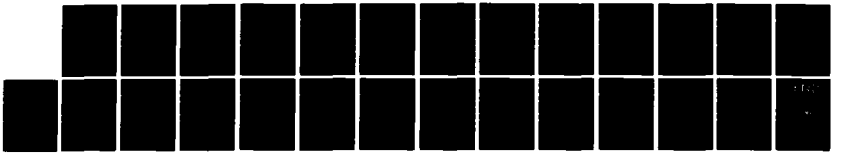
```
procedure comattr ( var rs:list1; attname:list2; var n1:integer;
                    m1:integer);


var k,n2 : integer;


begin
  n2:=n1;
  for i:=1 to m1 do
    begin
      k:=0;
      if  (attname[i]<>' ') then
        begin
          findattr(attname[i],rs,n2,k);
          if (k=1) then
          for j:=1 to n2 do
            rs[j].pr^.flag :='0';
          if (k>1) then
            begin
              n1:=n1+1;
              new(rs[n1].pr);
              new(rs[n1].ps);
              with rs[n1].pr^ do
                begin
                  keyattr[1]:=attname[i];
                  keyattr[2]:='*';
                  nonkeyattr[1]:='*';
                  flag := '0';
                end;     (* with *)
              with  rs[n1].ps^ do
                for j:=1 to mn do
                  tsets[j]:=0;
              for j:=1 to n2 do
                if (rs[j].pr^.flag='1') then
                  begin
                    deleteattr(attname[i],rs[j],pr^);
                    rs[n1].ps^.tsets[j] :=1;
                    rs[j].ps^.tsets[n1] :=-1;
                  end;  (* if *)
            end;    (* if *)
        end; (* if *)
    end;    (* for *)
end;   (* procedure comattr *) .
```

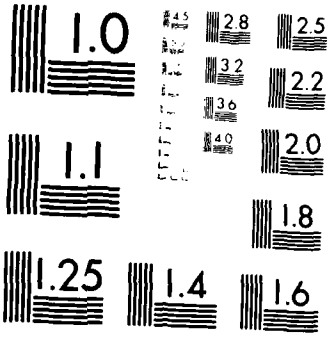
```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :                    9/9/84                     *)
(*     VERSION   :                      1.0                      *)
(*     NAME      :              Procedure systsets               *)
(*     MODULE NUMBER     :                5                      *)
(*     FUNCTION :     Crates a system owned set for each record  *)
(*                    which is not a member in any set.          *)
(*     INPUTS    :              rs.nl                            *)
(*     OUTPUTS   :              rs                               *)
(*     GLOBAL VARIABLES USED     :            rs.nl              *)
(*     GLOBAL VARIABLES CHANGED  :             rs                *)
(*     GLOBAL TABLES USED        :              _                *)
(*     GLOBAL TABLES CHANGED     :              _                *)
(*     FILES READ       :                     _                 *)
(*     FILES WRITTEN    :                     _                 *)
(*     MODULES CALLED   :                     _                 *)
(*     CALLING MODULES  :                Main program          *)
(*     AUTHOR    :                   Maj.Elsayed Yaser AL-      *)
(*     HISTORY :                          _                      *)
(*                                                               *)
(*****************************************************************)
```

```
procedure systsets ( var rslist:[ n1:integer  ;

var  z :integer;

begin
  for i:=1 to n1 do
    begin
      z:=0;
      with rs[i],ns^ do
        begin
          for j:=1 to n1 do
            if (tsets[j]=-1) then z:=z+1;
          if (z=0)  then   tsets[i]:=2;
        end;   (* with *)
    end;  (* for *)
end;   (* procedure systsets *)
```

AD-A151 844 DEVELOPING AN AUTOMATED TECHNIQUE FOR TRANSLATING A RELATIONAL DATABASE I..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. E Y ALY 2/2

UNCLASSIFIED DEC 84 AFIT/GCS/MATH/84D-4 F/G 9/2 NL





```
(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*     DATE      :                    12/9/84                    *)
(*     VERSION   :                      1.0                      *)
(*     NAME      :              Procedure results                *)
(*     MODULE NUMBER :                  7                        *)
(*     FUNCTION  :        Displays the network schema.           *)
(*     INPUTS    :             ns,nl,ml,attname                  *)
(*     OUTPUTS   :            displayed network schema           *)
(*     GLOBAL VARIABLES USED      :       ns,nl,ml,attname       *)
(*     GLOBAL VARIABLES CHANGED   :               -              *)
(*     GLOBAL TABLES USED         :               -              *)
(*     GLOBAL TABLES CHANGED      :               -              *)
(*     FILES READ       :                         -              *)
(*     FILES WRITTEN    :                         -              *)
(*     MODULES CALLED   :                         -              *)
(*     CALLING MODULES  :                  Main program         *)
(*     AUTHOR    :                   Maj.Elsayed Yaser Aly       *)
(*     HISTORY   :                          -                    *)
(*                                                               *)
(*****************************************************************)
```

```
procedure results ( rs:list1; attname:list2; n1,m1:integer);

var  y1,y2 : integer;

begin
  for i:=1 to 5 do
    writeln;
  writeln('        THIS IS THE EQUIVALENT NETWORK SCHEMA      ') ;
  writeln;
  writeln;
  writeln('          THE NO. OF RECORD TYPES IS ',n1) ;
  writeln;
  writeln;
  writeln('          THE NO. OF ATTRIBUTES    IS ',m1);
  writeln;
  writeln;
  write('       THE ATTRIBUTES ARE :    ');
  for i:=1 to m1 do
    write(attname[i],' ');
  for i:=1 to n1 do
  begin
     for j:=1 to 3 do
        writeln;
    writeln('        THIS IS THE RECORD TYPE NO.: ',i);
    for j:=1 to 3 do
        writeln;
    with rs[i].pr^ do
    begin
      write(' The Key attributes of this record are  ');
      j:=1;
      y1:=0;
      while(keyattr[j]<>'*') do
      begin
        if (keyattr[j]<>' ') then
        begin
          write(keyattr[j]);
          y1:=1;
        end;
        j:=j+1;
      end;  (* while *)
      if (y1=0)  then  write('NONE');
      for j:=1 to 3 do
          writeln;
      write(' The nonkey attributes of this record are  ');
      j:=1;
      y2:=0;
```

```
    while ( nonkeyattr[j]<>'*') do
    begin
      if ( nonkeyattr[j]<>' ') then
      begin
        write(nonkeyattr[j]);
        y2:=1;
      end;
      j:=j+1;
    end; (* while *)
    if (y2=0) then write('NONE');
    for j:=1 to 3 do
      writeln;
    if ((y1=0)and(y2=0)) then
      writeln(' This record type is a link between other records ')
  end;  (* with *)
  with rs[i],ps^ do
    for j:=1 to n1 do
    begin
      if (tsets[j]=1) then
      begin
        writeln;
        writeln;
        writeln('There is a set between record types no',i,' and no',j);
        writeln('The owner of this set is the record no',i);
        writeln('The member in this set is the record no',j);
      end;  (* if *)
      if (tsets[j]=-1) then
      begin
        writeln;
        writeln;
        writeln('There is a set between record types no',i,' and no',j);
        writeln('The owner of this set is the record no',j);
        writeln('The member in this set is the record no',i);
      end;  (* if *)
      if (tsets[j]=2) then
      begin
        writeln;
        writeln;
        writeln('This is a system owned record type ');
        writeln;
      end;   (* if *)
    end; (* for *)
  end; (* for *)
  writeln;
  writeln;
  writeln('  THIS IS THE END OF THE NETWORK SCHEMA  ');
end; (* procedure results *)
```

```
begin
                    (* MAIN PROGRAM *)

  checkinput(n1,m1,x1);
  if ( x1=0 ) then
    begin
      inputdata(rs,attname,n1,m1);
      createsets(c,rs,n1);
      removeattr(rs,n1);
      combattr(rs,attname,n1,m1);
      systsets(rs,n1);
      results(rs,attname,n1,m1);
  end;
end.
```

Appendix C

Resulted Network Schema

```
          WELCOME TO PROGRAM TRANSLATE


THE FUNCTION OF THIS PROGRAM IS TO TRANSLATE A RELATIONAL
SCHEMA INTO AN EQUIVALENT NETWORK ONE . THE PROGRAM IS
DESIGNED TO HANDLE A RELATIONAL SCHEMA WITH THE FOLLOWING
PARAMETERS :

MAX. NO. OF RELATIONS [m]  IS  10
MAX. NO. OF ATTRIBUTES [n] IS  10
MAX. NO. OF KEY_ATTR. [11] IS   5
MAX. NO. OF NONKEY_ATTR. [12] IS  9
MAX. NO. OF RECORD TYPES IN THE GENERATED NETWORK SCHEMA
[mn] IS 20 WHICH IS EQUAL TO m+n .



IF ANY OF YOUR SCHEMA PARAMETERS EXCEEDS THESE MAX. VALUES,
PLEASE MODIFY THE CORRESPONDING CONSTANTS AT THE DECLARATION
PART OF THE PROGRAM TO SUIT YOUR NEEDS .

                THANK YOU
```

THIS IS THE EQUIVALENT NETWORK SCHEMA

THE NO. OF RECORD TYPES IS 9

THE NO. OF ATTRIBUTES IS 9

THE ATTRIBUTES ARE : a b c d h j K i p

THIS IS THE RECORD TYPE NO.: 1

The Key attributes of this record are a

The nonKey attributes of this record are c

There is a set between record types no 1 and no 2
The owner of this set is the record no 1
The member in this set is the record no 2

There is a set between record types no 1 and no 4
The owner of this set is the record no 1
The member in this set is the record no 4

There is a set between record types no 1 and no 8
The owner of this set is the record no 8
The member in this set is the record no 1

THIS IS THE RECORD TYPE NO.: 2

The Key attributes of this record are h

The nonKey attributes of this record are NONE

There is a set between record types no 2 and no 1
The owner of this set is the record no 1
The member in this set is the record no 2

There is a set between record types no 2 and no 3
The owner of this set is the record no 3
The member in this set is the record no 2

There is a set between record types no 2 and no 5
The owner of this set is the record no 2
The member in this set is the record no 5

THIS IS THE RECORD TYPE NO.: 3

The Key attributes of this record are d

The nonKey attributes of this record are NONE

There is a set between record types no 3 and no 2
The owner of this set is the record no 3
The member in this set is the record no 2

There is a set between record types no 3 and no 9
The owner of this set is the record no 9
The member in this set is the record no 3

THIS IS THE RECORD TYPE NO.: 4

The key attributes of this record are i

The nonkey attributes of this record are NONE

There is a set between record types no 4 and no 1
The owner of this set is the record no 1
The member in this set is the record no 4

There is a set between record types no 4 and no 5
The owner of this set is the record no 4
The member in this set is the record no 5

There is a set between record types no 4 and no 9
The owner of this set is the record no 9
The member in this set is the record no 4

THIS IS THE RECORD TYPE NO.: 5

The key attributes of this record are p

The nonkey attributes of this record are NONE

There is a set between record types no 5 and no 2
The owner of this set is the record no 2
The member in this set is the record no 5

There is a set between record types no 7 and no 4
The owner of this set is the record no 4
The member in this set is the record no 7

There is a set between record types no 5 and no 7
The owner of this set is the record no 5
The member in this set is the record no 7

THIS IS THE RECORD TYPE NO.: 6

The Key attributes of this record are j

The nonKey attributes of this record are NONE

There is a set between record types no 6 and no 7
The owner of this set is the record no 6
The member in this set is the record no 7

There is a set between record types no 6 and no 8
The owner of this set is the record no 8
The member in this set is the record no 6

There is a set between record types no 6 and no 9
The owner of this set is the record no 9
The member in this set is the record no 6

THIS IS THE RECORD TYPE NO.: 7

The Key attributes of this record are NONE

The nonKey attributes of this record are NONE

This record type is a link between other records

There is a set between record types no 7 and no 5
The owner of this set is the record no 5
The member in this set is the record no 7

There is a set between record types no 7 and no 6
The owner of this set is the record no 6
The member in this set is the record no 7

THIS IS THE RECORD TYPE NO.: 8

The key attributes of this record are B

The nonkey attributes of this record are NONE

There is a set between record types no 8 and no 1
The owner of this set is the record no 8
The member in this set is the record no 1

There is a set between record types no 8 and no 6
The owner of this set is the record no 8
The member in this set is the record no 6

This is a system owned record type

THIS IS THE RECORD TYPE NO.: 9

The key attributes of this record are K

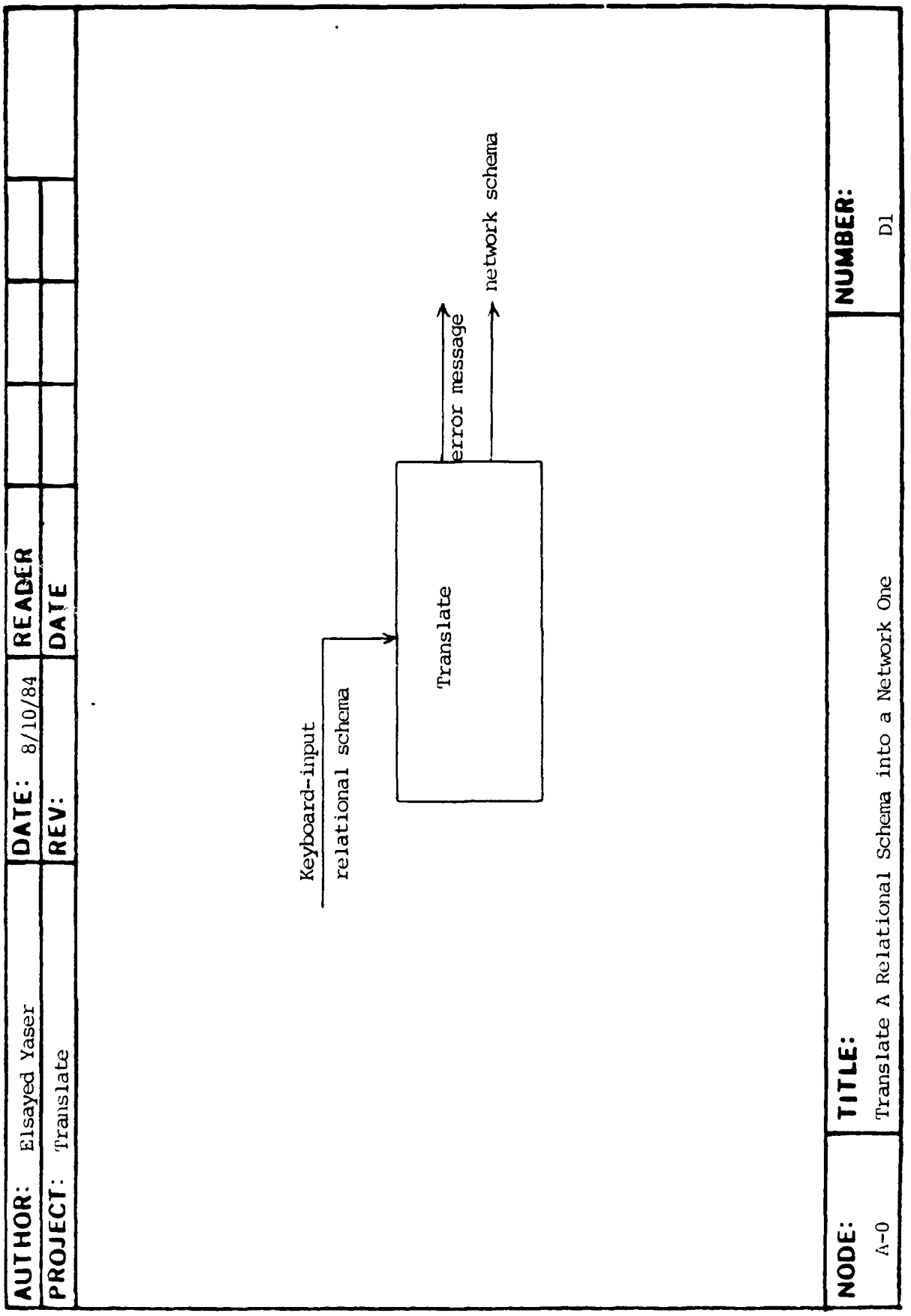
The nonkey attributes of this record are NONE

There is a set between record types no 9 and no 3
The owner of this set is the record no 9
The member in this set is the record no 3

There is a set between record types no 9 and no 4
The owner of this set is the record no 9
The member in this set is the record no 4

There is a set between record types no 9 and no 7
The owner of this set is the record no 9
The member in this set is the record no 7

This is a system owned record type

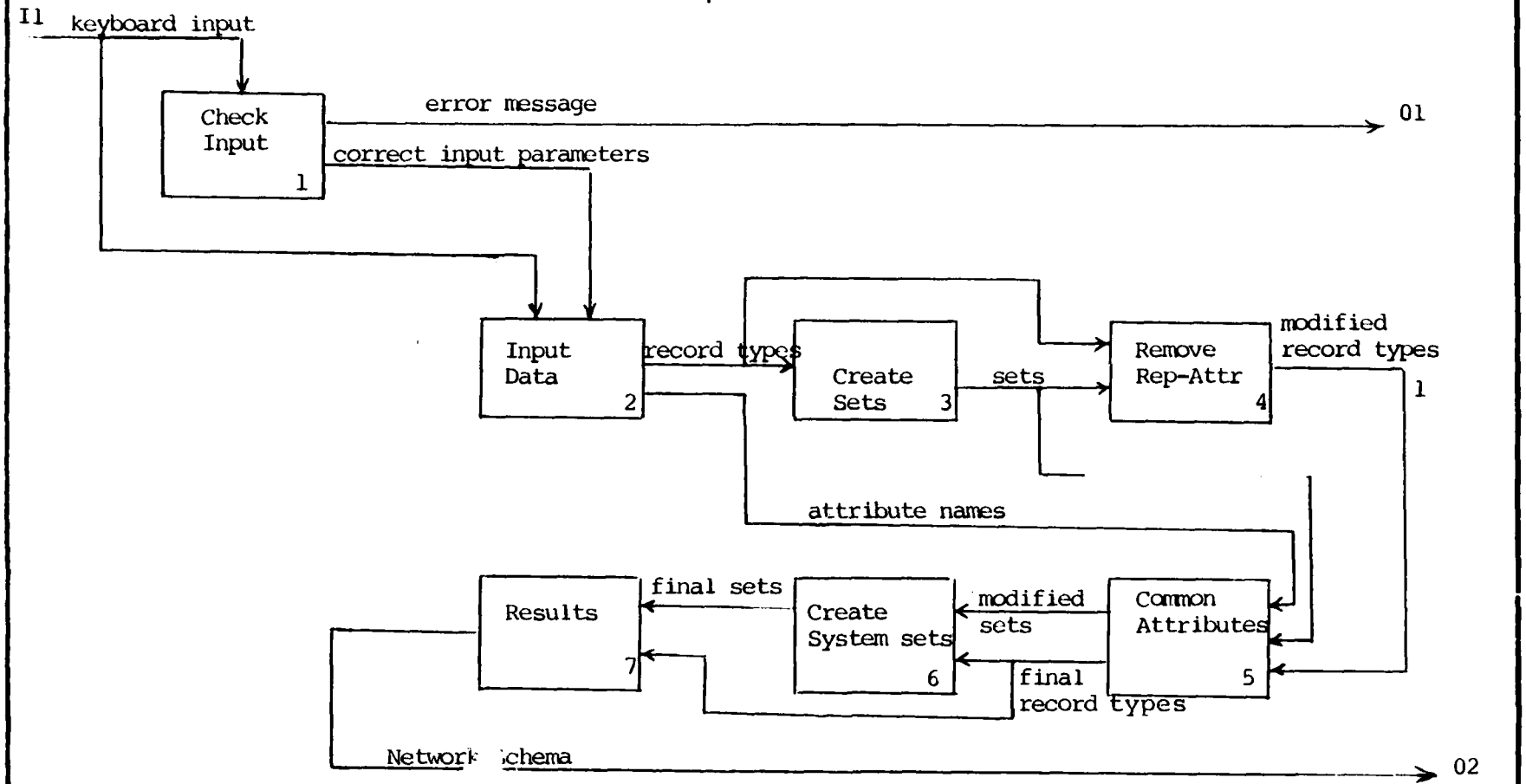

AUTHOR: Elsayed Yaser
PROJECT: Translate
DATE: 8/10/84
REV:
READER
DATE
NODE: A-0
TITLE: Translate A Relational Schema into a Network One
NUMBER: D1
Keyboard-input
relational schema
Translate
error message
network schema

A-0 TRANSLATE: Translate a Relational Schema into an Equivalent Network One

This is the entire software. It translates a relational schema into an equivalent network one. It is an interactive software, i.e., the user has to enter the relational schema as an input through the keyboard.

If there is an input error, an error message is displayed and the software stops the execution. In case of correct input data, the generated network schema is displayed on CRT and/or printed.

AUTHOR: Elsayed Yaser	DATE: 8/10/84	READER				
PROJECT: Translate	REV:	DATE				



NODE: A0	TITLE: Translate Translate a Relational Schema into a Network One	NUMBER: D2

A0 TRANSLATE: Translate a Relational Schema into an Equivalent Network One

The entire software is decomposed into the following seven major components:

1. Check input: It checks that the number of relations and the number of attributes of the relational schema are within the range of the software. If any of them exceeds the max. value, an error message is displayed and the software stops execution.

2. Input data: It takes the relational schema as its input and creates a record type for each relation.

3. Create sets: It creates the proper sets between the record types according to step 2 in the algorithm in section 4.1.

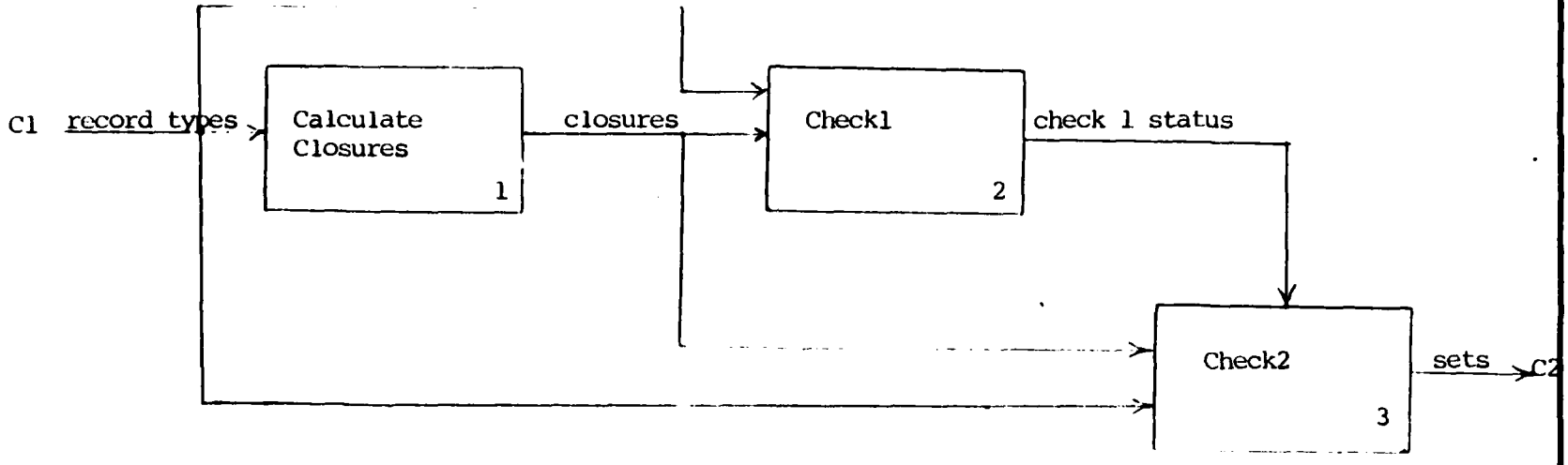
4. Remove repeated attributes: For every record type, it removes all the attributes that appear in its ancestors.

5. Common Attributes: For any common attribute, it creates a new record type consists of that attribute, deletes the attribute from the record types containing it, and creates the proper sets between the new record type and the other record types.

6. Create System Sets: It creates a system owned set for every record type that is not a member in any set.

7. Results: It produces the generated network schema in an understandable form.

AUTHOR: Elsayed Yaser	DATE: 8/10/84	READER				
PROJECT: Translate	REV:	DATE				



NODE: A3	TITLE: Create Sets	NUMBER: D3

A3 CREATE SETS

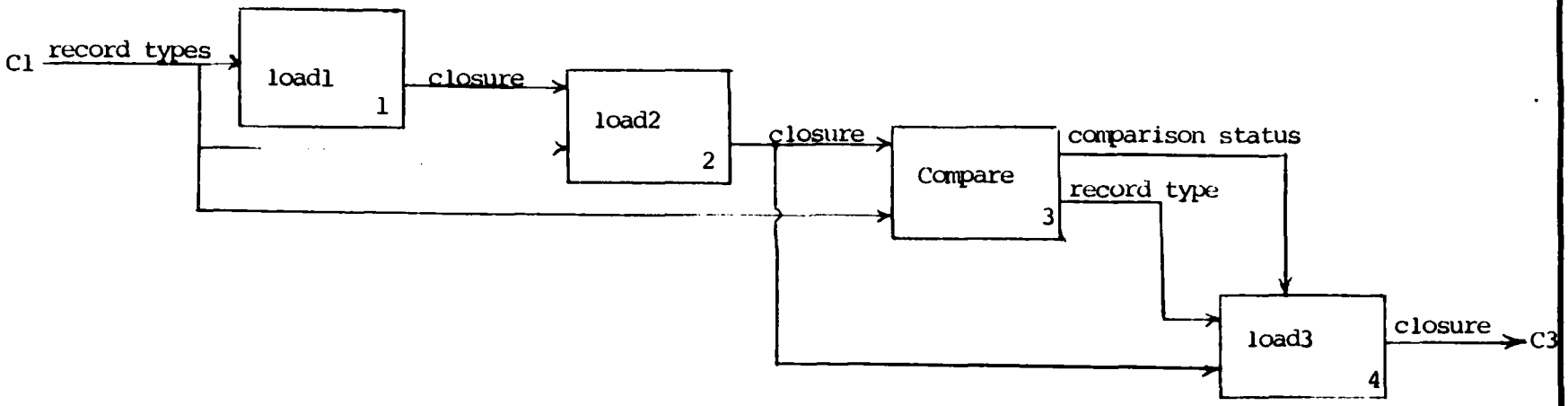
This part of the software creates the proper sets between the record types according to step 2 in the algorithm in 4.1. It consists of the following three components:

1. Calculate closures: It calculates the closure of each relation R_i corresponding to the record type N_i

2. Check1: For each pair R_i and R_j, this module checks that $R_i \subseteq R_j^+$ and there is no directed path from N_i to N_j. If these two conditions are correct, the module sets up the control variable check1 status.

3. Check2: This module is controlled by the variable check1 status. It checks that there is no R_k such that $R_i \subseteq R_k^+ \subseteq R_j^+$, and $K \neq j$. If this check is correct, the module creates a set such that N_i owns N_j.

AUTHOR: Elsayed Yaser	DATE: 8/10/84	READER				
PROJECT: Translate	REV:	DATE				



NODE:	TITLE:	NUMBER:
A31	Calculate Closures	D4

A31 CALCULATE CLOSURES

This part of the software calculates the closure of each relation in the relational schema. This can be done through the following steps:

1. load1: It loads the key attributes of each relation R_i in its closure R_i^+.

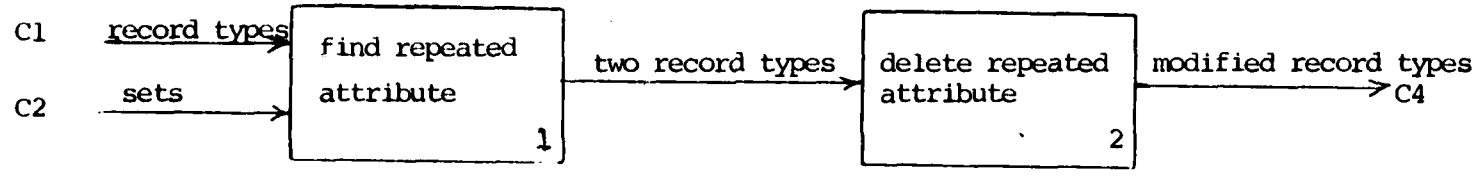
2. load2: It loads the nonkey attributes of each relation R_i in its closure R_i^+.

3. Compare: For each relation R_i, and for $j = 1,....,n$, $i \neq j$, this module checks that $R_j \subseteq R_i^+$. If this condition is satisfied, the module sets up the comparison status.

4. load3: This module is controlled by the comparison status variable If any attribute in R_j does not exist in R_i^+, the module adds it to R_i^+.

Steps 3 and 4 are repeated until it is sure that no new attributes will be added to R_i^+.

AUTHOR:	Elsayed Yaser	DATE: 8/10/84	READER				
PROJECT:	Translate	REV:	DATE				



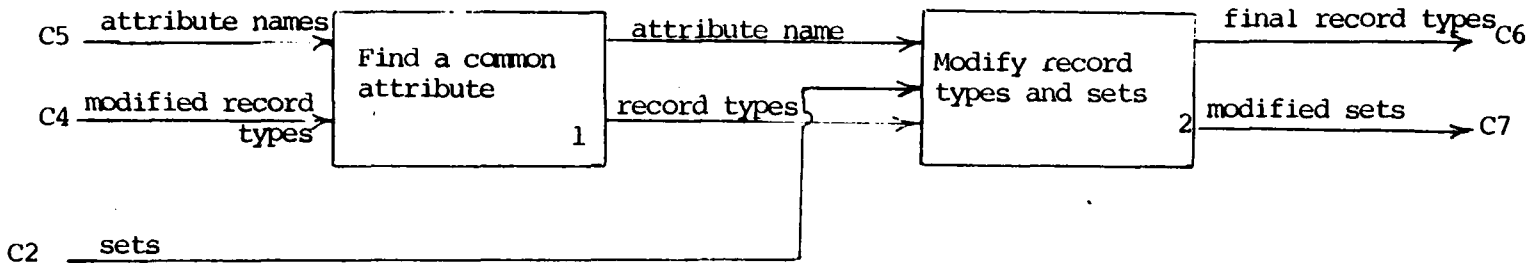
NODE:	TITLE:	NUMBER:
A4	Remove Repeated Attributes	D5

A4 REMOVE REPEATED ATTRIBUTES

For each record type N_i, this part of the software deletes all the attributes that appear in the ancestors of that record type. This is done in the following two steps:

1. find repeated attribute: This module looks for an attribute that is repeated between a record type and one of its ancestors.
2. delete repeated attribute: This module deletes the repeated attribute from the record type.

AUTHOR: Elsayed Yaser	DATE: 8/10/84	READER				
PROJECT: Translate	REV:	DATE				



NODE: A5	TITLE: Common Attributes	NUMBER: D6

A5 COMMON ATTRIBUTES

This part of the software executes the fourth step in algorithm 4.1. This step is done by two modules. They are:

1. Find a common attribute: For each attribute in the network schema, the module checks if it appears in more than one record type. In this case the module sets a flag in each record type contains that attribute.
2. Modify record types and sets

   For each common attribute, this module does the following:

   - create a new record type consists of that attribute
   - make the new record type an owner for every record type contains that attribute
   - delete the attribute from the old record types.

Vita

Maj Elsayed Yaser Aly was born on 27 September 1951 in Elsharkya, Egypt. He graduated from high school in Cairo, Egypt in 1969 and attended the Military Technical College, Cairo, from which he received the degree of Bachelor of Science in Electrical Engineering, branch of Computer Science in May 1974. Upon receiving his commission he was assigned to the Department of Weapons and Ammunition where he served in different positions as: electrical engineer in the maintenance and repair of special purpose military computers for two years; Chief engineer of an Air Defense Brigade for three years; and chief of planning and training branch in a Main Workshop for two years.

In 1980, he attended the Miltitary Technical College, from which he received a Diploma in Operations Research in December 1982. In 1981 he attended Ain Shams University, Cairo from which he received a Diploma in Computer Science in Commercial Applications in May 1983.

In 1983, he was selected to come to the USA, to attend the Air Force Institute of Technology, School of Engineering, to study for a Masters Degree in Computer Systems.

Permanent Address: Elsayed Yaser Aly
4 Asad St. Toumanbay St.
Elzaytoun, Cairo, Egypt

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/MATH/84D-4	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DCS/Computer Resources 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) HQ Electronic Security Command (Bldg 2000) Kelly AFB, San Antonio TX 78243	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11 TITLE (Include Security Classification) See Box 19	
12 PERSONAL AUTHOR(S) Elsayed Yaser Aly, Major, Egyptian Army	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1984 December 15. PAGE COUNT 113
16. SUPPLEMENTARY NOTATION	
17 COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Database, Multi-Model DBMS - Database Translation, Relational Database, Network Database

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: Developing an Automated Technique for Translating a Relational Database into an Equivalent Network One.

Thesis Chairman: Dr. Henry Potoczny

Approved for public release: IAW AFR 190-1
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Henry Potoczny	22b TELEPHONE NUMBER (Include Area Code) 513-255-3098 22c OFFICE SYMBOL AFIT/ENC

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

This thesis developed an automated technique for translating a relational schema into an equivalent network one. In addition to the usual translation applications, this technique is the major step towards developing the multi-model data base management system which enables implementing both relational and network databases via one system, allows the use of either relational or network commands, solves the problem of converting applications, and eases the communication between data base management systems.

Reviewing the previous efforts in that field, there are two algorithms that were developed for the same objective. Each one was discussed, explained, and used to solve an illustrative example. Both algorithms were compared and one of them was chosen as a basis for developing the automated technique.

The chosen algorithm was modified to suit the requirements, and analyzed and decomposed using the SADT, The automated technique was developed and coded in PASCAL. In its development, it was consisered to be user friendly, and to produce the network schema in a suitable format. The developed technique was tested and implemented using a relational schema in the third normal form. The resulting network schema was efficient, nonredundant, and had the minimum and necessary number of record types and sets as it was supposed to be.

Several conclusions were reached during the course of study, and some recommendations were proposed for further study according to the assumptions listed initially.

END

FILMED

5-85

DTIC